Simulation-based optimization for design parameter exploration in hybrid system: a defense system example

Jeong Hee Hong, Kyung-Min Seo and Tag Gon Kim

SIMULATION 2013 89: 362 originally published online 28 January 2013
DOI: 10.1177/0037549712466707

The online version of this article can be found at:
http://sim.sagepub.com/content/89/3/362

Published by:
http://www.sagepublications.com

On behalf of:

Society for Modeling and Simulation International (SCS)

Additional services and information for SIMULATION can be found at:

Email Alerts: http://sim.sagepub.com/cgi/alerts
Subscriptions: http://sim.sagepub.com/subscriptions
Reprints: http://www.sagepub.com/journalsReprints.nav
Permissions: http://www.sagepub.com/journalsPermissions.nav
Citations: http://sim.sagepub.com/content/89/3/362.refs.html

>> Version of Record - Apr 4, 2013
OnlineFirst Version of Record - Jan 28, 2013

What is This?
Simulation-based optimization for
design parameter exploration in hybrid
system: a defense system example

Jeong Hee Hong, Kyung-Min Seo and Tag Gon Kim

Abstract
This paper presents a method for solving the optimization problems that arise in hybrid systems. These systems are characterized by a combination of continuous and discrete event systems. The proposed method aims to find optimal design configurations that satisfy a goal performance. For exploring design parameter space, the proposed method integrates a metamodel and a metaheuristic method. The role of the metamodel is to give good initial candidates and reduced search space to the metaheuristic optimizer. On the other hand, the metaheuristic method improves the quality of the given candidates. This proposal also demonstrates a defense system that illustrates the practical application of the presented method. The optimization objective of the case study is to find the required operational capability configurations of a decoy that meet the desired measure of effectiveness. Through a comparison with a full search method, two metamodeling methods without the aid of metaheuristics and a metaheuristic method without the support of metamodels, we confirmed that the proposed method provides same high-quality solutions as those of the full search method at a small computational cost.

Keywords
simulation-based optimization, design parameter exploration, hybrid system, defense system, metamodel, metaheuristics

1. Introduction
Many applications in the manufacturing, economics, and military fields are called hybrid systems, because they exhibit both continuous and discrete dynamic behavior. The behavior of the hybrid system is characterized by interactions between continuous and discrete event systems (DESs). In most non-trivial cases, these hybrid systems are very complex and mathematically intractable, and they lack a closed-form expression for the objective function or the constraints. Simulation can be an appropriate tool for system analysis and design configuration optimization in this context. For example, a war game system, which is widely used for operating and analyzing defense systems, consists of various types of systems. To find the optimal specifications for the weapons that will achieve a goal performance, most war game systems have performed simulation evaluations of all possible parameter combinations, as a "what-if" analysis. It is an extremely time-consuming approach. On the other hand, simulation-based optimization provides a structured approach in order to find the optimal configurations that satisfy a performance goal, when analytical expressions of the relationships between input and output are unavailable. For this reason, simulation-based optimization techniques are frequently used for solving optimization problems for various complex systems.

Numerous simulation-based optimization studies have been proposed to find the best design configurations for complex systems. Metamodeling and metaheuristic methods are the most commonly used techniques in simulation-based optimization. Metamodel-based optimization transforms intractable problems into problems that can be solved. The implicitly representing stochastic response of the simulation as an explicit deterministic functional form makes this transformation possible. However, when the response surface is high-dimensional, non-differentiable, or discontinuous, the metamodeling technique sometimes fails to discover the optimal
solutions. On the other hand, metaheuristic methods are not problem-specific and make few or no assumptions.\(^9\) Metaheuristic methods, such as simulated annealing (SA),\(^{10,11}\) tabu search,\(^{12}\) and genetic algorithm (GA),\(^{13}\) are iteratively trying to improve the optimal candidates by exploring the search space. Nevertheless, metaheuristics can be ineffective and inefficient if the starting point is at a great distance from the optimal solutions. Many researchers have proposed various mixed strategies in economics,\(^{14,15}\) urban transportation,\(^{16–20}\) and manufacturing\(^{21–23}\) applications. They combined the advantages of both the metamodeling and metaheuristic methods, but all of them have used metamodels to filter candidates of metaheuristic methods, and have focused only on DESs.

Unfortunately, the classical simulation-based optimization works provide no further guidance as to how to conduct a search over the alternative design parameters of a hybrid system due to the behavior features of this system. The discontinuous dynamics of the hybrid system are caused by bounds on the discrete control events or sequences of the unequal state transition of DESs. If the state transition sequence of DESs is known, the optimal design parameter profiles may be found, but how can the optimal sequence be found without a combinatorial enumeration of all possible sequences depending on interaction with continuous systems (CSs)? In order to solve the computationally expensive and difficult optimization problems of hybrid systems, this paper presents a new strategy that we adapted from conventional simulation-based optimization concepts. To the best of our knowledge, this is the first paper in which a simulation-based optimization approach is used to solve the optimization problems of hybrid systems.

For exploring the design parameter space of the hybrid system, the proposed method integrates a metamodel and a metaheuristic method. The role of the metamodel is to accelerate the convergence of the metaheuristic method through good initial candidates that provide near-optimal solutions and reduced search space. On the other hand, the metaheuristic method improves the quality of candidates given by the metamodel. The optimization process for our method consists of three phases: preparatory, space partition, and local search phases. The preparatory phase defines the optimization objective, the parameters classification, and the stopping criteria. Then, the space partition phase divides the entire search space into disjointed subregions, constructs metamodels for the subregions, and identifies neighborhood zones, which include the feasible solutions using backward mapping of the metamodels. In other words, the purpose of incorporating the space partition is to reduce the simulation effort intelligently in the neighborhood zone selection. Selecting candidates close to the optimal solutions also leads to less simulation effort in the next phase. Finally, the local search phase intensively explores the selected neighborhood zones in order to improve candidates given in the selected neighborhood zones.

The proposed method is applied to an anti-torpedo warfare system in order to find the required operational capability (ROC) configurations of decoys that meet the desired measure of effectiveness (MOE). The anti-torpedo warfare system consists of engineering-level models, represented as CSs, and an engagement-level model, which serves as a DES. To evaluate the proposed method, we compared the proposed method with a full search, two metamodeling methods without the support of metaheuristics and a metaheuristic method without the support of metamodels. The comparison results show that the proposed method provides the same high-quality solutions as those of the full search method at a small computational cost.

This paper is organized as follows. Section 2 provides a review of several previous studies about simulation-based optimization. Section 3 outlines the optimization problems for the hybrid system. Section 4 introduces our proposed method for the hybrid system. The case study is described in Section 5 and a conclusion is given in Section 6.

### 2. Related works

This section reviews the most commonly used metamodeling and metaheuristic techniques in simulation-based optimization. We investigate the features and the drawbacks of each method. Then, we introduce the existing mixed approach, which combines the advantages of both techniques.

#### 2.1. Metamodeling method

The metamodeling technique is an approximation of the simulation model, which represents the relationship between design parameters and responses. The objective of metamodeling is to reduce the computational cost of the simulation model by replacing it. Two of the most useful approaches to metamodel construction are a statistics-based approach and a machine-learning approach.

The statistics-based approach depends on the data received from the simulation experiments. The regression models\(^{24}\) are commonly used in practice because of their manageable characteristics. Assis and Milani\(^{25}\) used piece-wise linear functions to develop optimal schedules for urban transportation, and Merkuryeva\(^{26}\) applied these to the optimization of production and logistics. In addition, Yalcinkaya and Mirac Bayhan\(^{19}\) tried to use the regression method to optimize the schedule of a metro line. Lu et al.\(^{27}\) developed a forward-inverse fitting method that used a recursive decomposition method to split the design region into invertible subregions, each with the response approximated by a different regression function. Barton\(^{28}\) represented a backward mapping method for design synthesis that breaks the design space into featured-based
subregions and then fitted the linear regression approximations. Nevertheless, these techniques are valid only within the range of the original data, and the fitting function is occasionally difficult to obtain, particularly in the case of a high-dimensional, non-differentiable, and discontinuous dynamics system (i.e., a hybrid system).

The machine-learning approach is based on neural networking, rule learning, and fuzzy logic. Only experimental data from simulations have been used to train the surrogate models. Neural network methods have been used in manufacturing, business, and military applications, due to the superiority of back-propagation. In some cases, they can provide more comprehensive and accurate solutions than the regression. However, insufficient training data and inappropriate model validation can yield inaccurate models. That is, building a good learning model often requires a high computational cost.

2.2. Metaheuristic method

The metaheuristic method is an iterative process that moves from current solutions to high-quality solutions. SA and GA methods are certainly the most popular optimization techniques in metaheuristics.

The SA method has been developed from statistical thermodynamics in order to simulate the behavior of atomic arrangements in liquid or solid materials during the annealing process. The SA method is widely utilized in optimization problems because of its inherent simplicity and its ability to find the optimal solution rapidly if a feasible starting point is given. Lambert also proposed a multi-level annealing strategy to escape from local optima. However, the SA method does not ensure that movement falls entirely within a feasible region at all times. Thus, the annealing procedure might require a long amount of time to reach a true optimum.

The GA is a tool for seeking solutions to the optimization problems of complex characteristics and large search spaces. The GA was inspired by the survival of the fittest principle, which was introduced by Charles Darwin on the subject of natural evolution. The driving force of the GA comes from a mixture of the reproduction (selection scheme) and recombination (crossover and mutation) mechanisms. Through the selection scheme, good candidates are reproduced and the self-adaptation that results from using the recombination mechanism prevents premature convergence. McHaney presented a simple real-coded GA to find better simulation parameters of the hard inverse problem. Al-Aomar intended to solve parameter estimation problems of the hybrid system and were trying to adopt efficient search techniques in metaheuristics. They common ideas are often very successful, because they manage, in some way or another, to combine the advantage of metaheuristic methods with the strength of metamodeling methods. The power of metaheuristic methods is certainly based on the concept of exploring solutions to obtain new trials, while a notable strength of metamodeling methods is that it reduces the simulation evaluation cost of new trials by filtering the trials. Persson and Hashemi combined a neural network with a hill-climbing algorithm, while Gholam et al. combined a neural network with a particle swarm optimization technique. April et al. and Syberfeldt et al. proposed a combination of a neural network and the GA. Merkuryeva et al. guided the search through the GA and improved GA solutions by using a response surface model (RSM)-based linear search.

Although metaheuristic methods have achieved great success in many applications, these algorithms have also encountered some technical hurdles. When specifically performing searches in the SA or GA, the question of how to generate the initial solution arises. Both of them commonly depend on the initial point, and they sometimes need a large number of simulation evaluations to achieve acceptable solutions. Even if the computer processing speed has been improved by Moore’s law, one single simulation evaluation may take from a couple of seconds to several days.

2.3. Mixed method

As noted above, both metamodeling and metaheuristic methods are powerful simulation-based optimization techniques, but both also have weaknesses under certain conditions. Fortunately, many researchers have been trying to overcome the drawbacks of each technique by using a combination of the metamodeling and metaheuristic methods. Their common ideas are often very successful, because they manage, in some way or another, to combine the advantage of metaheuristic methods with the strength of metamodeling methods. The power of metaheuristic methods is certainly based on the concept of exploring solutions to obtain new trials, while a notable strength of metamodeling methods is that it reduces the simulation evaluation cost of new trials by filtering the trials. Persson et al. integrated a neural network and a hill-climbing algorithm, while Gholam et al. combined a neural network with a particle swarm optimization technique. April et al. and Persson et al. proposed a combination of a neural network and the GA. Merkuryeva et al. guided the search through the GA and improved GA solutions by using a response surface model (RSM)-based linear search.

Previous researchers have been solely interested in a DES. To the best of our knowledge, there have been no attempts to use a simulation-based optimization approach for hybrid systems. We focused on the optimization problem for the hybrid system and were trying to adopt efficient techniques for the hybrid system.

3. Problem description

In this section, we describe our motivation for the optimization of the hybrid system. Simulation has become indispensable for the design of complex and combined discrete continuous systems for diverse applications. For example, a defense system, as a typical example of the hybrid system, is widely used to perform effectiveness analysis (MOE) and required weapon performance (MOP, measure of performance) analysis. Effectiveness and MOP analysis can be explained in forward simulation and reverse simulation, respectively. Figure 1 shows the concept for
forward simulation and reverse simulation. The goal of forward simulation of the defense system is to analyze system effectiveness by using a given MOP. The defense system consists of engagement-level and engineering-level models, as shown on the left-hand side of Figure 1. The engagement-level model includes a discrete decision process of a DES, while the engineering-level model is typically identified as a CS. By interacting with the engagement-level and engineering-level models, the defense system permits the evaluation of various scenarios or weapon specifications. On the contrary, forward simulation can be also used to find a weapons specifications MOP by using a given MOE. In such a case, repetitive simulations of all possible MOP combinations would be performed, which incurs a high computational cost. Such a trial-and-error process is not a cost-effective approach.

A reduction of such cost is a motivation for reverse simulation, in which the input and output are inverted in comparison to those in forward simulation, as shown on the right-hand side of Figure 1. Our fundamental goal is to seek optimal values for the MOP that satisfy a required MOE in a backward manner. Even if the concept of reverse simulation is proposed, no theory to execute a simulation model in such a backward manner is developed. However, the concept can be realized by means of optimization with the simulation model, which is called simulation-based optimization. Simulation-based optimization enables reverse simulation to be transformed into forward simulation. Our goal is to find an input such that the difference between a desired output and a forward simulation output is minimized in its absolute value. An objective function $f_{obj}$ for optimization is formulated as Equation (1):

$$ f_{obj} = \min |y^* - E[f(X_1, \ldots, X_n)]| $$

Constraints: $X^L_i \leq X_i \leq X^U_i$

$f(\cdot)$: hybrid simulation model

Design configuration MOP is represented as a set $X$ of the design parameters $X_i$, and the required MOE is represented as a desired output $y^*$. We want to find the optimal value of $X_i$, at which the difference between desired performance $y^*$ and the estimated value of the hybrid simulation model $f(X_1, \ldots, X_n)$ is minimized for all possible values of design variable $X_i$ within the range constraints $X^L_i \leq X_i \leq X^U_i$. Note that $X^L_i$ and $X^U_i$ are the lower and upper bounds for $X_i$. In general, we will need to use the sample mean over the $m$ replications of $f(X_1, \ldots, X_n)$ as an estimate of $E[f(X_1, \ldots, X_n)]$, since the hybrid simulation model $f(X_1, \ldots, X_n)$ is a stochastic model. Such stochastic characteristics may cause the simulation result of the same $X_i$ to be irreproducible. However, identical results can be obtained if the same random seed and the same number of replications are used.

**Figure 1.** Optimization problem of a hybrid system: a defense system example.
If we represent the hybrid model \( f(\cdot) \) as a metamodel \( \hat{f}(\cdot) \), then we can expect fast-running approximations, as well as a useful method for finding optimal design configurations for \( X_i \). The metamodel \( \hat{f}(\cdot) \) for the hybrid system plays a role in the development of inverse mappings. Since continuous states in the hybrid model are piecewise continuous in intervals divided by different discrete state transition patterns, the metamodel \( \hat{f}(\cdot) \) can be approximated to a set of piecewise functions. In addition, for simplicity, the metamodel \( \hat{f}(\cdot) \) is formed into a linear function. We believe that the linear approximation error can be compensated for by using a metaheuristic method as a feedback mechanism.

4. Proposed method

The purpose of optimization is to find the feasible design parameter configurations that satisfy the desirable performance requirement. This section provides a detailed introduction to the proposed simulation-based optimization process for the hybrid system.

4.1. Process overview

The optimization of hybrid systems, such as military or manufacturing systems, is more complicated by the fact that hybrid systems display both continuous and discrete event dynamics. The interaction between the continuous and discrete event dynamics in hybrid systems often causes non-smoothness and/or discontinuities in the continuous states, as well as different state transition patterns in the discrete event states. The search for the optimal design configurations is much more difficult, because changes in the hybrid system cause a huge combinatorial number of possible responses. At this time, simulation-based optimization is useful, but previous works have only focused on DESs. Although the existing simulation-based optimization methods have not considered the hybrid system, our approach is similar to the concept of these classical simulation-based optimization processes, including a choice of candidate and a search. Besides focusing on the hybrid system, the difference between our approach and the previous approach is that the metamodeling method and the metaheuristic method support each other and compensate for each other’s weak points. The metamodel assists the selection of the good initial candidate for the metaheuristic method, and the metaheuristic method improves the quality of the candidate given by the metamodel.

As shown in Figure 2, the proposed method is composed of three phases: a preparatory phase, a space partition phase, and a local search phase. The preparatory phase sets up a goal performance, an objective function, the parameters classifications, and the stopping criteria. The goal of optimization is to find a setting of controllable parameters that minimizes the given objective function. Furthermore, this phase defines the fixed values of the uncontrollable parameters and the stopping criteria for the convergence condition of the metaheuristic search.

The space partition phase consists of designing an experiment to divide the overall design space into subregions, and building the metamodel for each subregion. To reduce the computational burden of the optimization process, we adopted a metamodeling method, since the computational cost associated with running a metamodel is negligible in comparison to the cost of simulation runs. However, the metamodel construction for the hybrid system is not straightforward, because the response of the hybrid system involves different dynamics behavior in intervals of the design parameters. The dynamics of the hybrid system, caused by interactions between the CS and the DES, is changed at some points. Such points identify the boundaries of each subregion. A collection of the different dynamics in each subregion is characterized by the relationship between input parameters and the corresponding response. Thus, a metamodel for the hybrid system would be constructed in a piecewise manner. For simplicity, we approximate the metamodel to a linear regression model. The cost of the metamodel construction can be greatly reduced if a good sample set is selected by the design of experiment (DOE), and if the number of simulation runs and replications are clearly determined. After the simulation experiments are complete, the overall design space is divided into disjointed subregions and formulated into a linear form based on the simulation results.

The last local search phase improves the current point by tracing neighbors around the optimal point. Even if fitting the metamodel in each subregion is validated, subregion models are not guaranteed to completely represent the system because of a linear approximation. In this case, local improvement helps the search to thoroughly explore the selected neighborhood zones that are near a good candidate. Any kind of metaheuristic search methods can be used for local improvement. The candidate obtained by the metaheuristic method is compared with an existing candidate, and the winner becomes the new solution. This competition is repeated until the stopping criterion is satisfied. In other words, although an initial optimal candidate provided by the previous phase is not the optimal solution, the iterative improvement by the metaheuristic method can move it to the optimal solution. In the following sections, we will explain each phase in detail.

4.2. Preparatory phase

In the preparatory phase, we specify the objective of optimization, controllable parameters, and the stopping criteria, as summarized in Table 1.
Table 1. Terminology used in the preparatory phase.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Controllable parameters ($X_c$)</td>
<td>Parameters that are adjustable by the system designers</td>
</tr>
<tr>
<td>Uncontrollable parameters ($X_u$)</td>
<td>Parameters that are unadjustable and preset</td>
</tr>
<tr>
<td>Qualitative parameters ($X_Q$)</td>
<td>Parameters that are described in terms of some quality or categorization</td>
</tr>
<tr>
<td>Quantitative parameters ($X_Q$)</td>
<td>Parameters that are measured or identified on a numerical scale</td>
</tr>
<tr>
<td>Stopping criteria ($c$)</td>
<td>Condition upon which a process for finding optimal solutions stops</td>
</tr>
</tbody>
</table>
The design parameters $X$ are divided into two sets: those that are controllable and those that are uncontrollable (i.e., $X = X_c \cup X_u$). The controllable parameters $X_c$ are closely related to the optimization objective. The purpose of optimization is to find the best values of $X_c$ that produce the goal performance (i.e., the output of the system). It is assumed that the uncontrollable parameters $X_u$ are preset as the operational point of the system. The design parameters $X$ are also classified into two types: qualitative $X_{ql}$ and quantitative $X_{qn}$. We assume that quantitative parameters $X_{qn}$ are defined within lower and upper bounds, and qualitative parameters $X_{ql}$ describe items in terms of some quality or categorization. In the next space partition phase, dummy variables can be defined as $X_{dp}$ incorporated into metamodel construction such that it assumes the value 1 whenever the category it represents occurs. Otherwise, the value 0 is assumed, because $X_{ql}$ can be binary or categorical data.

As previously mentioned, our optimization process is complete when the difference between a targeted response and a simulation output of the optimal candidate is minimized in its absolute value. Of course, the value would be zero in an ideal case, which is practically impossible because of the computational cost. Therefore, we find the solution that satisfies the stopping criteria, instead of looking for the best one according to the stopping criteria. However, if the value of the stopping criteria is small enough, the solution is good enough, even though it may not be the best. In this case, the solution involves choosing either a configuration with the optimum value of the objective function, or a configuration whose objective function value is close to the optimal value. Normally, the stopping criteria are some predefined computational conditions, such as the length of computation time, the number of iterations, and/or the acceptable gradient $\varepsilon$. The difference between response $f(\bar{x})$ of the optimal candidate and desirable performance $y^*$ is used as the stopping rule: $|y^* - f(\bar{x})| < \varepsilon$. If $\varepsilon$ is too large, the local search process stops earlier than it should before the true optimal point is found. On the other hand, if $\varepsilon$ is too small, the target values fluctuate in the neighborhood of the optimal point. We choose to use the gradient $\varepsilon$ for the stopping criteria. Other computational time and the number of iterations are examined for evaluating the performance of the proposed method.

### 4.3. Space partition phase

The space partition phase divides the overall design space into subregions according to the behavioral characteristics of the hybrid system. Then, they are represented as an easily comprehensible analytic model through a metamodeling technique. Some subregions are selected as neighborhood zones, including feasible solutions.

#### 4.3.1. Design of experiment

The first step of the space partition phase is to sample data in the entire design space. DOE techniques are effective at reducing the computational expense of constructing high-quality metamodels. Various DOE techniques have been studied for planning a series of trials without any restriction of system types. The most popular DOE methods are central composite (CC) and Latin hypercube (LH) designs.

The CC design is the most widely used for the RSM. Although the rotatability of CC design is a desirable property for the RSM, it does not necessarily extend the points within the design variable bounds. For example, the rotatable design $\alpha$ value is set to 1.4, which has extended the design region beyond the variable bounds of the defined design. Since the face-centered central composite (FCC) design can be used when the region of operability encompasses the region of interest, as defined by the variable bounds, we have adopted it. The FCC design consists of all vertices, the center of all faces, and the center of the design space.

In addition, with the assumption of a linear model, Ye developed an algorithm for orthogonal LH design that was extended by Cioppa, who exchanged small amounts of non-orthogonality for better space-filling and developed the nearly orthogonal Latin hypercube (NOLH) design. The NOLH design possesses improved space-filling properties with nearly orthogonal parameters for LH design. We also add the NOLH design to our DOE step.

#### 4.3.2. Partition of disjointed subregions

This step is the crucial part in the overall optimization process. Figure 3(a) shows the state of a hybrid system that combines the CS and the DES. State transition of the DES complies with internal and external transition rules, and those of the CS follow a differential equation. The state of the hybrid system is described by the values of continuous states and discrete control modes according to the interactions between the CS and DES. The state changes of both systems influence those of each other. As shown in Figure 3(b) or (c), a discrete event changes the relation of the continuous state variable, while a CS variable reaching a threshold causes a discrete event. Through the comparison of state transition dynamics in Figure 3(b) and (c), the changes in input parameters, such as $P_c$, lead to different state transition dynamics sets between the CS and the DES. Figure 3(b) shows the dynamics of the hybrid system in $v_1 \leq P_c < v_2$, and (c) displays the dynamics in $v_2 \leq P_c < v_3$. The different dynamics, depending on the value of $P_c$, bring a piecewise continuous relationship between an input parameter $P_c$ and an output $Y$, as shown in Figure 3(d). A jump in $v_2$ or $v_3$ results from changes in the complexity of continuous behavior and the corresponding discrete control mode. The entire design space is divided into disjointed subregions with respect to jump points in each input parameter. For
the simplicity of the metamodel construction, we assume that the behavior in each range of the input parameter is linearized.

According to the state transition characteristics of the hybrid system, the design space \( S = \bigcap_{k=1}^{n_c} [X_k^L, X_k^U] \subseteq \mathbb{R}^{n_x} \) of dimension \( n_c \) is partitioned by \( n_R \) disjointed subregions \( Z_i \): \( \bigcup_{i=1}^{n_R} Z_i = S \), \( Z_i \cap Z_j = \emptyset \) for \( i \neq j \). Here, \( X_k \) is an element of controllable parameters \( X_c \) and \( n_x \) is the size of \( X_c \). Each subregion \( Z_i \) is linked to an appropriate model that is used for selecting an optimal candidate. Using the simulation results of inputs sampled in the DOE step, we can identify the jump points for the range partition of parameters. Figure 4 shows an example of the design space partitioning. Let the design space \( S \) be a two-dimensional space: \([X_{1_{\text{min}}}, X_{1_{\text{max}}}] \times [X_{2_{\text{min}}}, X_{2_{\text{max}}}]\), \( n_x = 2 \). For \( X_1 \), we can recognize that the response increases in \( X_{1_{\text{min}}} \leq X_1 \leq X_{1_{\text{cut}}} \) and decreases in \( X_{1_{\text{cut}}} \leq X_1 \leq X_{1_{\text{max}}} \). The response function contains a jump discontinuity at \( X_{1_{\text{cut}}} \). In the same way, the cutting point \( X_{2_{\text{cut}}} \) is obtained. Eventually, the design space is divided into four subregions \( \{Z_1, \ldots, Z_4\} \).

### 4.3.3. Metamodel construction for subregions.

In this step, the response for each subregion \( Z_i \) is approximated to a linear regression metamodel \( \hat{f}_i \). The metamodel of the overall design space is represented by a collection of piecewise linear functions in an explicit and comprehensible form. The locally linear model for each subregion facilitates backward mapping between the desired performance \( y^* \) and the optimal design candidate \( x^* \). R-square, the Durbin–Watson statistic, multicollinearity, and the \( p \)-value of the coefficients are checked for the validation of the constructed linear regression metamodels. As shown in Figure 5, each

---

**Figure 3.** State transition of a hybrid system. (a) State of the hybrid system. (b) \( v_1 \leq P_c < v_2 \). (c) \( v_2 \leq P_c < v_3 \). (d) \( v_1 \leq P_c < v_3 \).

**Figure 4.** Example of partitioning disjointed subregions in two-dimensional design variables \((X_1, X_2)\).
metamodel for each subregion $Z_i$ is constructed on a flat surface, because the projection of $X_1$ or $X_2$ is a linear function.

4.3.4. Neighborhood zone selection. This step is used to find all of the qualified subregions that satisfy the desired performance. The linear function $\hat{f}(\cdot)$ in each subregion $Z_i$ is determined within the range of its upper bound $Z_{i\text{max}}$ and lower bound $Z_{i\text{min}}$, because of bounded design parameters. Figure 6 shows an example of the process. For a certain given value of performance $y^*$, the subregion $Z_2$, with its value falling in between its performance range, $Z_{2\text{min}} \leq y^* \leq Z_{2\text{max}}$, is selected as the neighborhood zone, as shown in Figure 6(a). Eventually, it narrows the overall design space down to subregion $Z_2$, as shown in Figure 6(b).

4.4. Local search phase

The local search phase includes the selection of optimal candidates from the metamodels of the promising neighborhood zones, and touches on how to improve the quality of candidates obtained. Even if the selected neighborhood zones contain optimal solutions, metamodels of them do not always provide optimal solutions on the first try because of the approximate manner in which metamodels are incorporated into subregions. The high-quality solutions are achieved through an intensive exploration around neighbors.

4.4.1. Optimal candidate selection. This process selects optimal candidates by converting linear metamodels $\hat{f}(\cdot)$ of selected neighborhood zones into inverse metamodels $\hat{f}^{-1}(\cdot)$. Figure 7 explains how to select an optimal candidate from the selected subregion $Z_2$ in the previous example, as depicted in Figure 6. Suppose that we are trying to obtain the candidate of $X_1$. In the selected subregion $Z_2$, a linear function $f_2(X_1)$ is projected on $X_1$, as shown in
Figure 7(a). Inputting the desired performance \( y^* \) into the inverse model \( f^{-1}_2(x_1) \) produces the optimal candidate \( \hat{x}_1 \), as we can see in Figure 7(b). If multiple neighborhood zones are selected, multiple candidates might be selected.

4.4.2. Local improvement. In this step, an iterative exploration can come close to the optimal solution by using a metaheuristic search optimizer. For the search optimizer, any metaheuristic method can be applied to this step. An example of metaheuristic methods is a SA algorithm. In our case study, SA is simple and useful because of a good starting point given by the previous step. Figure 8 continues the example from Figure 7 for the reader’s benefit. In Figure 8(a), the targeted performance \( y^* \) is compared to the evaluation result of optimal candidate \( \hat{x}_1 \). If the difference \( \delta, |y^* - f_2(x_1)| \) does not satisfy the stopping criteria \( \varepsilon \), the metaheuristic method iteratively finds the next candidate \( \hat{x}_1' \) until the stopping criterion is satisfied, as shown in Figure 8(b).

5. Case study

The proposed simulation-based optimization method is applied to an anti-torpedo warfare model. In general, underwater warfare can be represented by a hybrid system, including various complex underwater weapon systems, such as torpedoes, decoys, submarines, and warships. We abstracted that the system consists of a friendly warship with four decoy systems, and an enemy torpedo as a threat. The anti-torpedo warfare model employs the survival rate analysis of the warship, depending on its defense strategies or the performance of its decoy systems. The objective of optimization is to find the optimal design configurations of the decoy system for a desired survival rate, in a given overall design space. In the model, input parameter sets become tactics and MOPs that are called ROCs, and output becomes a MOE. The next section explains the anti-torpedo warfare scenario and then performs the proposed optimization process for finding the ROC of the decoy that is satisfied with the given MOE.

5.1. Anti-torpedo warfare scenario

A brief scenario for the combat is described in Figure 9.

1. An enemy torpedo is launched from an enemy submarine and searches out targets using its searching algorithm.
2. A friendly warship detects that an enemy torpedo is heading toward it when the torpedo enters the warning range.
3. The warship activates its defense strategy:
   3.1 The warship activates its decoy systems, according to operational tactics, deployments, and motion types;
3.2 after firing the decoys, the warship detours to evade the torpedo in accordance with evasion tactics.

4. The torpedo will mistake the decoys for the warship and attack one of the decoys.

5. The torpedo will search for the warship again.

Given the simplified combat situation, we have clearly made some assumptions. Firstly, the friendly warship already knows the performance of a torpedo that has been fired from an enemy submarine. Secondly, the defense strategies are limited to the deployment, launch type, and motion type of the decoys as well as the detour directions of the warship. Finally, the warship has four decoy systems.

The anti-torpedo warfare model consists of engagement- and engineering-level models. The engagement-level model, as a DES, controls the overall simulation of the anti-torpedo warfare model. This model plays an important role in the tactical operation and the survival rate analysis. On the other hand, the engineering-level model, as a CS, is typically a detailed mathematical representation of individual weapon systems, such as the maneuvering of the warship, the torpedo, and decoys. Modeled major entities are illustrated in Table 2.

### 5.2. Optimization procedure

The proposed optimization process is applied to find the ROC configurations of the decoys for the targeted MOE.

#### 5.2.1. Preparatory phase. As previously mentioned, this phase classifies design parameters and formulates the objective function and the stopping criteria.

**Step 1. Parameter definition and classification**

Our anti-torpedo warfare system has 10 design parameters, as illustrated in Table 3. We define three parameters ($P_{\text{WR}}$, $P_{\text{T}}$, $P_{\text{EC}}$) for the warship, two parameters ($P_{\text{D}}$, $P_{\text{T}}$) for the torpedo, and five parameters ($P_{\text{DP}}$, $P_{\text{LT}}$, $P_{\text{d}}$, $P_{\text{V}}$, $P_{\text{SL}}$) for the decoy. Parameters are classified into...

---

**Table 2. Anti-torpedo warfare system descriptions.**

<table>
<thead>
<tr>
<th>Entity name</th>
<th>System type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EA</td>
<td>DES</td>
<td>Decision-making of engagement orders against an incoming enemy torpedo and collector/analyzer results (survival rate, MOE)</td>
</tr>
<tr>
<td>Warship</td>
<td>CS</td>
<td>A maneuvering part of the friendly warship and warning against an enemy torpedo</td>
</tr>
<tr>
<td>Decoy</td>
<td>CS</td>
<td>A weapon system for defending against an enemy torpedo</td>
</tr>
<tr>
<td>Torpedo</td>
<td>CS</td>
<td>An enemy threat launched by an enemy submarine</td>
</tr>
</tbody>
</table>

EA: effectiveness analysis.
those that are controllable and uncontrollable, as well as qualitative and quantitative. Each quantitative parameter has its lower and upper bound, while qualitative parameters are modeled to categorical data. The parameters for the decoy are controllable, because the purpose of our optimization is to find the optimal design configuration of the decoy using the given MOE. The remaining parameters are considered uncontrollable parameters that are assigned to the default setting value.

Step 2. Objective function

Let the targeted MOE* be 65%, and the controllable parameters be $P_{DP}$, $P_{LT}$, $P_d$, $P_{V_d}$, and $P_{SL}$. The uncontrollable parameters $P_{WR}$, $P_{V_c}$, $P_{EC}$, $P_{V_c}$, and $P_{TM}$ are set to 4000 m, 15 knots, 90°, 35 knots, and $TM_1$, respectively. Our objective is to find optimal configuration sets of $P_{DP}$, $P_{LT}$, $P_d$, $P_{V_d}$, and $P_{SL}$ for achieving a 65% survival rate. The anti-torpedo warfare model is defined as Equation (2), and the objective function in Equation (1) is substituted for Equation (3). The stopping criteria $c$ is set to 5 and the number of replications for the expectation of $f(\cdot)$ is assigned to 100. In addition, the anti-torpedo warfare system is stochastic, and it can make the output irreproducible. The same random seed is used in the case study in order to avoid this:

$$y = MOE = f(P_{DP}, P_{LT}, P_d, P_{V_d}, P_{SL})$$

$$f_{obj} = \min [65 - E[f(P_{DP}, P_{LT}, P_d, P_{V_d}, P_{SL})]]$$

constraints: $P_{DP} = \{DP_1, DP_2, DP_3, DP_4\}$,
$P_{LT} = \{LT_1, LT_2\}$,
$150 \leq P_d \leq 540$ (interval: 30),
$6 \leq P_{V_d} \leq 15$ (interval: 3),
$120 \leq P_{SL} \leq 180$ (interval: 20),
$f(\cdot)$: hybrid simulation model

5.2.2. Space partition phase. The design space partition phase is carried out over the following three steps.

Step 1. Design of experiment

In our five-dimensional controllable parameters, the full factorial design size is 4 (for $P_{DP}$) $\times$ 2 (for $P_{LT}$) $\times$ 14 (for $P_d$) $\times$ 4 (for $P_{V_d}$) $\times$ 4 (for $P_{SL}$) = 1792, which

<table>
<thead>
<tr>
<th>Related entity</th>
<th>Name</th>
<th>Classification</th>
<th>Description</th>
<th>Implication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Warship</td>
<td>Warning range of warship ($P_{WR}$)</td>
<td>$X_o/X_{qn}$</td>
<td>[2000, 5000] (m) interval: 500</td>
<td>The radius for detecting threats</td>
</tr>
<tr>
<td>Speed of warship ($P_{V_c}$)</td>
<td>$X_o/X_{qn}$</td>
<td>[15, 30] (knots) interval: 3</td>
<td>The top velocity of the warship</td>
<td></td>
</tr>
<tr>
<td>Evasive course ($P_{EC}$)</td>
<td>$X_o/X_{qn}$</td>
<td>[30, 150] (°) interval: 10</td>
<td>The detour direction of the warship</td>
<td></td>
</tr>
<tr>
<td>Torpedo</td>
<td>Speed of torpedo ($P_{V_c}$)</td>
<td>$X_o/X_{qn}$</td>
<td>[15, 30] (knots) interval: 30</td>
<td>The top velocity of the torpedo</td>
</tr>
<tr>
<td>Torpedo mode ($P_{TM}$)</td>
<td>$X_o/X_{ql}$</td>
<td>$TM_1$</td>
<td>Fire-and-forget: torpedo that does not require further guidance after launch</td>
<td></td>
</tr>
<tr>
<td>Decoy</td>
<td>Decoy deployment pattern ($P_{DP}$)</td>
<td>$X_o/X_{ql}$</td>
<td>$DP_1$</td>
<td>4 static decoys</td>
</tr>
<tr>
<td>Decoy launch type ($P_{LT}$)</td>
<td>$X_o/X_{ql}$</td>
<td>$LT_1$</td>
<td>Rocket</td>
<td></td>
</tr>
<tr>
<td>Operating duration of decoy ($P_d$)</td>
<td>$X_o/X_{qn}$</td>
<td>$[150, 540]$ (s) interval: 30</td>
<td>Time from launch to expiration</td>
<td></td>
</tr>
<tr>
<td>Speed of decoy ($P_{V_d}$)</td>
<td>$X_o/X_{qn}$</td>
<td>$[6, 15]$ (knots) interval: 3</td>
<td>The top velocity of the mobile decoy</td>
<td></td>
</tr>
<tr>
<td>Source level of decoy ($P_{SL}$)</td>
<td>$X_o/X_{qn}$</td>
<td>[120, 180] (dB) interval: 20</td>
<td>The level of decoy-generated noise</td>
<td></td>
</tr>
</tbody>
</table>

Table 3. Design parameters for our anti-torpedo warfare system.

$X = \{X_1, X_2, \ldots, X_{10}\} = \{P_{WR}, P_{V_c}, P_{EC}, P_{TM}, P_{LT}, P_d, P_{V_d}, P_{SL}\}$
requires a large number of experiment sets and significant computational time. The average execution time for a single case is 0.16 hours, and the total computational time for 1792 cases is 309.88 hours. As explained earlier, the sample points were reduced from 1792 to 44 by using a FCC design (27 points), and a NOLH design (17 points). The computational time was also reduced from 309.88 hours to 8.44 hours.

Step 2. Subregions partitioning and metamodel construction

The 44 samples obtained are used as inputs for the anti-torpedo warfare model to generate the output MOE. After inspecting the marginalized outputs at the individual parameter, each parameter was divided into a disjointed range, as shown in Figure 10. The overall design space is partitioned into eight subregions by means of combining the range of each parameter. Table 4 shows the disjointed subregions and their metamodels, which represent the relationship between the design parameter space and the performance space. For example, a subregion $Z_1$ is defined as a linear regression metamodel 

$$ f_1 = 152.95 + 42.11P_{dm_1} - 27.73P_{dm_2} + 16.54P_{dm_3} - 15.14P_{dm_4} - 0.23P_d + 1.96P_v - 0.48P_{SL} \text{ under } 150 \leq P_d < 240, 6 \leq P_v < 12, \text{ and } 120 \leq P_{SL} < 160. $$

For constructing the linear regression metamodel, dummy variables $P_{dm_1}$, $P_{dm_2}$, $P_{dm_3}$, $P_{dm_4}$ are used for qualitative parameter $P_{DP}$, and $P_{dm_5}$ is used for $P_{LT}$. The output value in subregion $Z_1$ exists between 6 ($Z_{1_{min}}$) and 100 ($Z_{1_{max}}$). The fitness of the metamodels for the subregions is evaluated by $R$-square, the Durbin–Watson statistic, multicollinearity, and the $p$-value of the coefficients. Table 4 illustrates the adjusted $R$-square and the $p$-value of the metamodels.
Step 3. Neighborhood zone selection

After successfully incorporating metamodels for eight subregions, we have to find all neighborhood zones among the overall subregions, including the feasible solutions that satisfy the targeted MOE* . Such means that the given MOE* falls between the lower bound $Z_{\text{inf}}$ and the upper bound $Z_{\text{sup}}$ of the targeted neighborhood zone $Z_i$. Given MOE* (65), we can select subregions $Z_1, Z_2, Z_3, Z_4, Z_6$ as feasible neighborhood zones according to the last column of Table 4. Selected neighborhood zones need to be investigated by each $P_{DP}$ parameter due to the encoding of dummy variables for the $P_{DP}$ parameter. Upon further examination, as shown in Table 5, $Z_1$ and $Z_5$ are finally chosen as neighborhood zones.

5.2.3. Local search phase. In this phase, the linear approximation errors of our metamodels are compensated by using a metaheuristic optimizer. SA was applied as an example of one metaheuristic method in the case study. Various metaheuristic methods can be used for improving the candidate.

Step 1. Optimal candidate selection

The optimal candidate is determined by an inverse mapping of each selected neighborhood from a desired MOE* (65). The obtained candidates have to lie within constraint of each selected subregion. For example, the candidates $(P_{DP}, P_{LT}, P_d, P_{Y_2}, P_{SL})$ selected in neighborhood $Z_i$ should meet Equation (4). As shown in Table 6, selected subregions $Z_1$ and $Z_5$ generate 12 and 20 candidates, respectively:

$$65 - \hat{f}_1(P_{DP}, P_{LT}, P_{Y_2}, P_{V_2}, P_{SL}) < 5 \tag{4}$$

$$P_{DP} = \{DP_1, DP_2, DP_3, DP_4\},$$

$$P_{LT} = \{LT_1, LT_2\},$$

$$150 \leq P_d < 240 \text{ (interval: 30)},$$

$$6 \leq P_{Y_2} < 12 \text{ (interval: 3)},$$

$$120 \leq P_{SL} < 160 \text{ (interval: 20)}$$

$$(P_{DP}, P_{LT}, P_d, P_{Y_2}, P_{SL}) \in \{(DP_3, LT_1, 150, 6, 140),$$

$$(DP_3, LT_1, 180, 6, 120), (DP_3, LT_1, 180, 9, 140),$$

$$(DP_3, LT_1, 210, 6, 120), (DP_3, LT_1, 210, 9, 120),$$

$$(DP_3, LT_2, 210, 6, 140), (DP_4, LT_1, 150, 9, 120),$$

$$(DP_4, LT_2, 150, 6, 140), (DP_4, LT_2, 150, 9, 140),$$

$$(DP_4, LT_2, 180, 6, 120), (DP_4, LT_2, 180, 9, 140),$$

$$(DP_4, LT_2, 210, 9, 120)\}$$

Step 2. Local improvement

To improve the quality of our solutions, the iterative search moves from the current solution to one of its neighbors in each selected subregion. In this case study, SA
the \(P_{DP}, P_{LT}, P_d, P_{V_i}, P_{SL}\) configurations that satisfy the targeted \(\text{MOE}^*\) with 65\%. For instance, \(P_{DP} = DP_3, P_{LT} = LT_1, P_d = 150, P_{V_i} = 6, P_{SL} = 140\), which was obtained from the optimal candidate selection in subregion \(Z_1\), is evaluated by the simulation, and then two additional iterations by SA move it to \(P_{DP} = DP_4, P_{LT} = LT_1, P_d = 150, P_{V_i} = 6, P_{SL} = 120\), which meets the stopping criteria. Other candidates are improved in the same way. Five candidates among the given 32 optimal candidates present optimal solutions that satisfy the stopping criteria, and the remaining 27 candidates go through local improvement. After local improvement, all 32 candidates are improved to 26 optimal solutions. Several candidates converge to the same optimal solutions.

### Table 5. Neighborhood zone selection.

<table>
<thead>
<tr>
<th>Subregion (Z_i)</th>
<th>Respond bound ([Z_{min}, Z_{max}]) at (P_{DP})</th>
<th>Select?</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Z_1)</td>
<td>([76, 100])</td>
<td>([6, 50])</td>
</tr>
<tr>
<td>(Z_4)</td>
<td>([74, 86])</td>
<td>([78, 90])</td>
</tr>
<tr>
<td>(Z_5)</td>
<td>([32, 100])</td>
<td>([24, 97])</td>
</tr>
<tr>
<td>(Z_6)</td>
<td>([91, 100])</td>
<td>([25, 36])</td>
</tr>
<tr>
<td>(Z_8)</td>
<td>([67, 82])</td>
<td>([72, 86])</td>
</tr>
</tbody>
</table>

### Table 6. Optimal candidate selection and evaluation and local improvement for the optimization of our anti-torpedo warfare system.

\(\text{MOE}^* = 65\%\): stopping criteria \(i = 5\)

Uncontrollable parameters \(P_{\text{WR}} = 4000\text{ m}; P_{V_i} = 15\text{ knots}; P_{EC} = 90\text{°}; P_{V_i} = 35\text{ knots}; P_{TM} = TM_1\)

<table>
<thead>
<tr>
<th>Neighborhood</th>
<th>Optimal candidate</th>
<th>Opt. candi. evaluation</th>
<th>Local improvement (LI)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Z_1)</td>
<td>(P_{DP}) (LT_1) 150 6 140 15</td>
<td>(P_{DP}, LT_1, 150, 6, 120) 63 2 0.40</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(P_{DP}) (LT_1) 150 6 120 59</td>
<td>(P_{DP}, LT_1, 180, 9, 120) 68 4 1.23</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(P_{DP}) (LT_1) 180 9 140 85</td>
<td>(P_{DP}, LT_1, 180, 9, 120) 68 1 0.46</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(P_{DP}) (LT_1) 210 6 120 60</td>
<td>(P_{DP}, LT_1, 210, 9, 140) 69 1 0.31</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(P_{DP}) (LT_1) 210 9 120 65</td>
<td>(P_{DP}, LT_1, 210, 9, 140) 65 0 0.18</td>
<td></td>
</tr>
<tr>
<td>(Z_2)</td>
<td>(P_{DP}) (LT_1) 210 9 120 54</td>
<td>(P_{DP}, LT_1, 150, 6, 120) 63 9 1.55</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(P_{DP}) (LT_1) 300 6 140 75</td>
<td>(P_{DP}, LT_1, 360, 6, 120) 65 4 0.91</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(P_{DP}) (LT_1) 330 6 120 78</td>
<td>(P_{DP}, LT_1, 360, 6, 120) 65 3 0.77</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(P_{DP}) (LT_1) 360 6 120 65</td>
<td>(P_{DP}, LT_1, 360, 6, 120) 65 0 0.24</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(P_{DP}) (LT_1) 360 9 140 33</td>
<td>(P_{DP}, LT_1, 360, 6, 120) 65 3 0.74</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(P_{DP}) (LT_1) 390 9 120 55</td>
<td>(P_{DP}, LT_1, 450, 6, 120) 69 19 3.33</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(P_{DP}) (LT_1) 360 6 120 65</td>
<td>(P_{DP}, LT_1, 360, 6, 120) 65 0 0.17</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(P_{DP}) (LT_1) 360 9 120 80</td>
<td>(P_{DP}, LT_1, 360, 6, 120) 65 9 1.96</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(P_{DP}) (LT_1) 450 9 140 33</td>
<td>(P_{DP}, LT_1, 510, 6, 140) 69 4 0.77</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(P_{DP}) (LT_1) 270 9 120 48</td>
<td>(P_{DP}, LT_1, 510, 6, 120) 62 2 0.59</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(P_{DP}) (LT_1) 270 6 140 90</td>
<td>(P_{DP}, LT_1, 270, 9, 140) 68 8 1.71</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(P_{DP}) (LT_1) 300 6 120 87</td>
<td>(P_{DP}, LT_1, 360, 6, 120) 65 8 2.59</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(P_{DP}) (LT_1) 330 9 140 96</td>
<td>(P_{DP}, LT_1, 330, 6, 140) 69 7 2.15</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(P_{DP}) (LT_1) 360 9 140 16</td>
<td>(P_{DP}, LT_1, 330, 6, 140) 69 7 3.47</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(P_{DP}) (LT_1) 510 6 140 69</td>
<td>(P_{DP}, LT_1, 510, 6, 140) 69 0 0.13</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(P_{DP}) (LT_1) 240 6 140 100</td>
<td>(P_{DP}, LT_1, 240, 6, 120) 67 3 0.71</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(P_{DP}) (LT_1) 270 6 120 99</td>
<td>(P_{DP}, LT_2, 240, 6, 120) 67 6 1.47</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(P_{DP}) (LT_1) 330 9 140 74</td>
<td>(P_{DP}, LT_1, 360, 6, 120) 65 6 1.22</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(P_{DP}) (LT_1) 360 9 120 92</td>
<td>(P_{DP}, LT_1, 360, 6, 120) 66 3 0.69</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(P_{DP}) (LT_1) 510 6 140 69</td>
<td>(P_{DP}, LT_1, 510, 6, 140) 69 0 0.15</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(P_{DP}) (LT_1) 540 6 120 77</td>
<td>(P_{DP}, LT_1, 510, 6, 140) 69 4 1.35</td>
<td></td>
</tr>
</tbody>
</table>

2 subregions 32 optimal candidates 26 solutions, total iter and time for LI: 135, 36.18

MOE: measure of effectiveness.
5.3. Analysis of the results

We compared the performance of our proposed method with other optimization techniques from the perspective of the quality of solutions and search speed. As previously mentioned, there is no competing optimization method for a hybrid system. Thus, we considered the full search method for evaluating quality of optimization solutions, because the full search method can always find all optimal solutions through the estimation of all possible input combinations. We also need to compare our approach with metamodel-based methods and a metaheuristic method, because our proposed method is combined. In our case study, we applied a piecewise linear metamodel and a SA algorithm. Thus, we compared a standard linear metamodel without a piecewise concept, a piecewise linear metamodel without the support of a metaheuristic method, and SA without the aid of a metamodel.

Figure 11 and Table 7 describe the quality of the solutions and the search speed of each method. Although our method incurs a small computational cost (211 iterations, 44.62 hours) compared with the full search method (1792 iterations, 309.88 hours), as shown in Table 7, our method is able to produce the same high-quality solutions as those of the full search method (see Figure 11). Such findings prove that our method performs more efficiently than the full search method.

Metamodel-based methods are very quick in terms of their search speed, due to their simplicity. The standard linear metamodel, however, generates poor solutions that lie far from the optimal point. As shown in Figure 11, 45 out of 47 solutions of the standard linear metamodel without the piecewise concept are beyond the range of the stopping criteria. Even if solutions obtained by the piecewise linear metamodel are also low quality, the piecewise linear metamodel provides better solutions than the standard linear metamodel. Hence, we confirmed that the piecewise concept is suitable for the hybrid system. We used

Table 7. Search speed comparison of full search, standard linear metamodel, piecewise linear metamodel, simulated annealing (SA), and proposed method.

<table>
<thead>
<tr>
<th>Method</th>
<th>Computational cost</th>
<th>Execution time (hour)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full search</td>
<td>$1792 \times 2\times 14\times 4 \times 4$</td>
<td>309.88</td>
</tr>
<tr>
<td>Metamodel (w/o piecewise)</td>
<td>$91 \times 44\times 32$</td>
<td>16.06 (8.44 + 7.62)</td>
</tr>
<tr>
<td>Metamodel (w/ piecewise)</td>
<td>$76 \times 44\times 32$</td>
<td>14.25 (8.44 + 5.81)</td>
</tr>
<tr>
<td>Simulated annealing (SA)</td>
<td>$735 \times 26\times 709$</td>
<td>142.29</td>
</tr>
<tr>
<td>Proposed method (Metamodel + Metaheuristic)</td>
<td>$211 \times 44\times 32\times 135$</td>
<td>44.62 (8.44 + 36.18)</td>
</tr>
</tbody>
</table>

Search speed: Metamodel (w/ piecewise) > Metamodel (w/o piecewise) > Proposed method > SA > Full search

MM: metamodeling; LI: local improvement.
such solutions of the piecewise linear metamodel as optimal candidates. In our proposed method, the quality of optimal candidates was dramatically improved by additional local improvement (135 iterations, 36.18 hours). This means that incorrect solutions of the piecewise linear metamodel finally moved to high-quality solutions. In comparison with the piecewise linear metamodel, we also verified the effectiveness of the local adjustment process. Despite the fact that we required the additional cost beyond the metamodel-based method, such an extra cost was much cheaper than the full search method or SA.

As shown in Figure 11, the solutions of SA without the assistance of a metamodel have the same high quality as the proposed method. However, SA requires much more computational cost (735 iterations, 142.29 hours) than our method (211 iterations, 44.62 hours), as shown in Table 7. The difference between SA and our method is whether or not there is the support of a metamodel. The search speed depends on the choice of the initial points and the size of the search space. SA uses randomly selected initial points, while our method uses candidates given by the piecewise linear metamodel. In other words, the initial points provided by the metamodel can be closer to the optimal solutions than the random selection. In addition, our method narrows the search space down to some subregions through neighborhood zones selection, while SA actually explores the entire search space. It can also help to reduce the search cost, as explained by the number of iterations and execution time.

As a result, our proposed method ensures the same high-quality solutions as those of the full search, but it does not cost a great deal more than the metamodel methods, as summarized in Figure 11 and Table 7.

6. Conclusion

This paper proposes a simulation-based optimization method for a hybrid system, while previous simulation-based optimization studies have focused on DESs. Our proposed method is a combination of a piecewise metamodel and a metaheuristic optimizer. The metamodel assists with the selection of the good initial candidate for the metaheuristic optimizer. In addition, the introduced piecewise concept, which represents the piecewise continuous behavior characteristics of the hybrid system, reduces the size of the search space by dividing the entire search space into disjointed subregions. However, metamodels of selected subregions do not always provide high-quality candidates, because of the approximate manner in which metamodels are constructed for subregions. Such an approximation error is compensated by the metaheuristic optimizer. The metaheuristic optimizer improves the quality of the candidates given by selected metamodels and rapidly obtains the high-quality solutions by using candidates that are close to the optimal solutions and reduced search space. In our method, the metamodel and the metaheuristic method are complementary.

This method was applied to an anti-torpedo warfare model, which consists of engagement- and engineering-level models in order to find the MOP configurations of decoys that meet the desired MOE. Various metaheuristic methods are available for the metaheuristic optimizer. For example, SA is used for employing local improvement. For evaluating the proposed method, we compared the proposed method with a full search, a standard linear regression metamodel, a piecewise linear regression metamodel without the support of metaheuristic methods, and SA without the support of metamodels. The results of the comparison show that our method provides the same high-quality solutions as those of the full search method, despite a small computational cost.

Although our method successfully solved the optimization problem of a five-variable system in the case study, we need to apply this approach to more complex and larger problems. There are also various metamodels or metaheuristic methods that can be considered. Furthermore, the metamodel can be used to filter candidates of the metaheuristic optimizer before evaluating them in the simulation model. Therefore, the performance of optimization strategies should be compared according to the way in which the metamodel is used. Although our proposed method fortunately found all optimal solutions in the case study, it might miss some optimal solutions due to wrong neighborhood zone selection. This is caused by approximation errors of the metamodel. We expect that our method can be extended to such studies in the near future.

Funding

This work was supported by Defense Acquisition Program Administration and Agency for Defense Development, Korea (contract UD110006MD).

References

conference on systemics, cybernetics and informatics (WMSCI 2010), 2010, pp.133–137.
43. Persson A, Grimm H and Ng A. Metamodel assisted global search using a probing technique. In: proceedings of the
IAENG international conference on artificial intelligence and applications (ICAIA’07), 2007.

Author biographies

Jeong Hee Hong is a PhD candidate at the department of electrical engineering at the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea. She received her BS in electrical engineering from Pusan National University, Busan, Korea, in 2005 and MS from the school of electrical engineering and computer science, KAIST, in 2007. Her research interests include simulation-based optimization, discrete event systems modeling/simulation, hybrid simulation, and distributed simulation.

Kyung-Min Seo received his BS in electrical engineering from Pusan National University and his MS in electrical engineering and computer science from KAIST in 2006 and 2008, respectively. Currently, he is a PhD candidate at the system modeling simulation laboratory (SMSLAB) at KAIST. His research interests include defense modeling and simulation and discrete event systems modeling/simulation.

Tag Gon Kim received his PhD in computer engineering with specialization in systems modeling and simulation from University of Arizona, Tucson, AZ, 1988. He was an assistant professor at the electrical and computer engineering department, University of Kansas, Lawrence, KS, from 1989 to 1991. He joined the electrical engineering department at KAIST, Daejeon, Korea, in fall, 1991, and has been a full professor at the electrical engineering and computer science (EECS) department since fall, 1998. He was the president of the Korea Society for Simulation (KSS) and the editor-in-chief for Simulation: Transactions for Society for Computer Modeling and Simulation International (SCS). He is a co-author of the text book, Theory of Modeling and Simulation, Academic Press, 2000. He has published about 200 papers in modeling and simulation theory and practice in international journals and conference proceedings. He is very active in defense modeling and simulation in Korea. He was/is a consultant for defense modeling and simulation at various Korean government organizations, including the Ministry of Defense, the Defense Agency for Technology and Quality (DTAQ), the Korea Institute for Defense Analysis (KIDA), and the Agency for Defense Development (ADD). He developed a toolset, called DEVSimHLA, for high-level architecture-compliant war game models development, which has been used for the development of three military war game models for the Navy, Air Force, and Marine force in Korea. He is a fellow of the SCS, a senior member of the Institute of Electrical and Electronics Engineers (IEEE) and Eta Kappa Nu.