AN APPROACH FOR COMBINED SIMULATION BASED PARAMETER AND STRUCTURE OPTIMIZATION USING EVOLUTIONARY ALGORITHMS

Olaf Hagendorf[1], Thorsten Pawletta[1], Christina Deatcu[1], Roland Larek[2]

[1]Hochschule Wismar, University of Applied Sciences: Technology, Business and Design
[2]Foundation Institute of Materials Science, Bremen

olaf.hagendorf@hs-wismar.de (Olaf Hagendorf)

Abstract

Modeling and simulation with integrated parameter optimization is used routinely to improve system performance. In this established technique model structure is considered to be fixed as the relationships between model elements are defined during model development. As model performance is optimized it may be necessary to redesign the model structure. The redesign is normally carried out manually.
Evolutionary Algorithms are a subtopic of Artificial Intelligence that are involved in combinatorial optimization problems. These algorithms are based on ideas inspired by biological evolution: reproduction and selection, mutation and recombination. They often perform well for many problem types because they do not make assumption about the problem specific search space.
The research reported in this paper details an approach providing optimization through automatic reconfiguration of both: model structure and model parameters. An evolutionary algorithm based optimization method is assisted by model management using a meta-modeling method. It searches for an optimal solution with repeated, combined model parameter and model structure changes resulting in a combined parameter and structure optimized model. Therefore the model management provides algorithms to join an evolutionary algorithm with model generation and simulation.

Keywords: DEVS, Structure Optimization, Evolutionary Algorithm, Parallel Computing.

Presenting Author's biography

Olaf Hagendorf studied electrical engineering with the specialization computer engineering at Universities Wismar and Rostock. After finishing his study in 1997 he set up a company, among other specialized in automation system and machine control development for the photofinishing industries. His company dealt with orders mainly in Middle and Northern Europe. Parallel to his business he started with a PhD project at Liverpool John Moores University and successfully finished it at 2009. Currently he is a research assistant at University Wismar.

1 Introduction

Modeling and simulation with integrated optimization is a well established technique in engineering applications. Such techniques are used for system design, real time planning and to control production systems. With increasingly complex, flexible production systems the requirements for modeling and simulation tools are growing. Existing applications using simulation based optimization are restricted to parameter optimizations. The user has to change model structure manually and repeat optimizations until a suitable solution is found. With increasing production system flexibility the number of possible structure variants increases and the potential benefit of automatic model structure optimization becomes significant.

The focus of this paper is the description of a methodology for a combined parameter and structure optimization for modular, hierarchical discrete event systems. In contrast to current modeling and simulation environments with integrated optimization the model structure is variable and thus it is open to optimization as well. The variations of model structure and model parameter values are controlled by a super ordinate optimization module. To support the optimization method appropriate modeling, model management/generation and simulation methods are necessary.

As a basis for model management and generation the System Entity Structure/Model Base (SES/MB) formalism, introduced by Rozenblit, Zeigler et al [7] [11] [12], is employed. The SES formalism is a generic, knowledge base framework consisting of a tree like entity structure and a model base. With its features the framework is able to define a set of modular, hierarchical models and to generate specific model structures. The modeling and simulation method is based on the Discrete Event System Specification (DEVS) formalism introduced by Zeigler [11] and some of its extensions [1], [3], [5] and [10].

Section 2 provides a short optimization and simulation based optimization overview, the new approach using a combined, simulation based structure and parameter optimization method is presented and the requirements for the optimization environment are defined. Section 3 briefly introduces the established SES formalism as a model set organization and model generating meta-modeling method. In section 4 the principles and prerequisites of evolutionary algorithms are discussed. The synthesis of the three elements, optimization, model generation and simulation, to perform a combined structure and parameter optimization, is presented in section 5. Finally an application to prove the concept is described in section 6.

2 Simulation Based Optimization

Simulation experiments can be of different complexity. The least complex ones are ordinary simulation runs, shown in figure 1a. After examining simulation results the user manually changes the model parameter values and/or structure and starts the simulation again. These steps are repeated until a suitable solution is found. A more complex approach is simulation based parameter optimization, described in figure 1b.

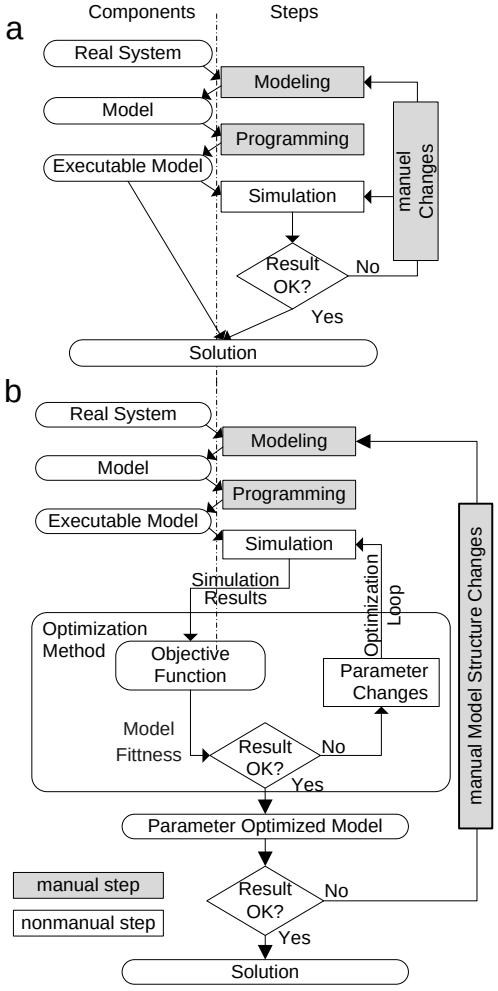

Fig. 1 Principles of (a) simulation and (b) simulation based parameter optimization

Mathematical optimization generally means establishing a function minimum or maximum. Simulation based parameter optimization means finding the optimal model input parameter value set by optimizing a function of output variables, named objective function and estimated with a simulation method [9]. The optimization method alters model parameter values to improve the result of the objective function until a stop criterion is fulfilled. The result is a parameter optimized model. Structure changes are carried out manually by a user followed by a repetition of the automated parameter optimization.

According to [9], a simulation based parameter optimization problem O with a set of m input

parameters $X = \{x_1, \ldots x_m\}$ can be formally described as follows:

- The parameter set $X = \{x_1, \ldots x_m\}$ with the domain set $D = \{d_1 \ldots d_m\}$.
- The multidimensional (one for each parameter) search space S defined by $S = \{s = \{(x_1,v_1) \ldots (x_m,v_m)\} \mid v_i \in d_i\}$
- The output set Y is defined by $Y = \{y_1 \ldots y_n\} = Y(X)$ and estimated by simulation.
- The objective function F establishes a single stochastic value from output set Y: $F = F(Y(X)) \rightarrow \Re+$ which is a measure of the current model performance
- Because of the stochastic nature of Y and consequently of F an estimation function R, the simulation response function, defined by $R(X) = E(F(Y(X)))$, is optimized

Each parameter value set $X_i \in S$ can be seen as a possible solution of O. The optimizer has to search the search space S to find the parameter value set $X_{opt} \in S$ with $E(F(Y(X_{opt}))) \leq E(F(Y(X_i))) \forall X_i \in S$. The resulting parameter value set X_{opt} is considered the global optimum of O.

It is important to note that automatic structure changes during optimization are not possible with this approach. Instead, structure changes are carried out manually by a user and each manual structure change requires a repetition of the automated parameter optimization. The idea behind this paper is the extension of the optimization method with the ability to change the model structure thus improving the objective function result. The effect of this extension is a simulation based structure and parameter optimization. Figure 2 presents the approach in principle. In contrast to the established approach:

1. This approach combines three methods: (i) a meta-model framework for model management, (ii) a modeling and simulation environment and (iii) an optimization method.
2. The optimization method controls both: the model parameter values and the model structure, changing both until a stop criterion is fulfilled. The result of this process is a combined parameter and structure optimized model.
3. The user has to organize a set of models. One possibility is to define a model which describes a set of model variants instead of one single model of the real system. Such models that define the creation and interpretation of a set of models are named meta-models. Through this inclusion of the meta-model based automatic model generating element the optimizer can additionally control model structure changes to find an optimized solution.

The extension of the formal description of a simulation based parameter optimization problem O to a combined simulation based structure and parameter optimization leads to O^*:

- The model parameter set X_P and its domain set D_P, above defined as X and D, are extended by structure parameter set X_S and its domain set D_S. The extended set definitions are: $X^* = X_P \cup X_S = \{x_{P1} \ldots x_{Pm}, x_{S1} \ldots x_{Sn}\}$ and $D^* = D_P \cup D_S = \{d_{P1} \ldots d_{Pm}, d_{S1} \ldots d_{Sn}\}$ with m model parameters in set X_P and n structure parameters in set X_S. The sets X_P and D_P are defined by the current model. The model management has to provide the sets X_S and D_S by analyzing the meta-model.

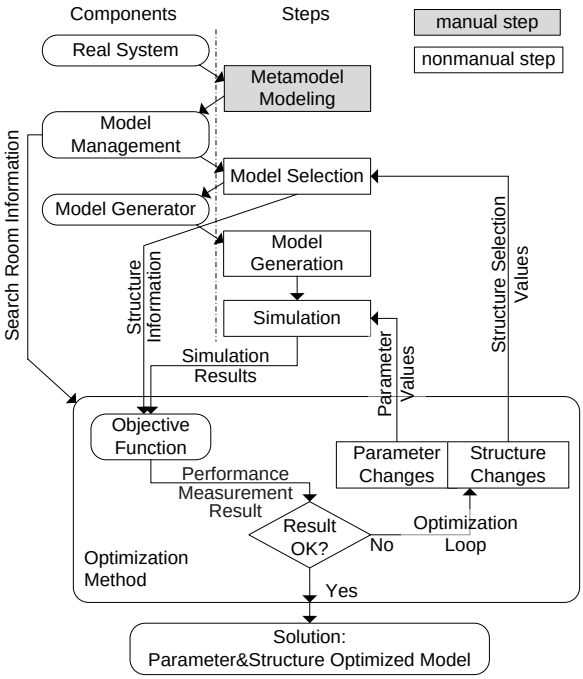

Fig. 2 Principle of a structure and simulation based parameter optimization

- The multi-dimensional (one for each parameter) search space $S = S_P \cup S_S$ is spanned by sets of model parameter and structure variants.
- The objective function F^* is defined by $F^*(Y(X^*),P(X_S))$ with simulation results $Y(X^*)=Y(X_S \cup X_P)$ and results based on structure related variables $P(X_S)$ which are established during the model selection. Because of the stochastic nature of the simulation results $Y(X^*)$ an estimation function R, the simulation response function, is calculated. The results based on structure related variables $P(X_S)$ are not stochastic. Hence, the simulation response function is defined by $R(Y(X^*))$ and subsequently the objective function by $F^*(R(Y(X^*)), P(X_S))$.

Crucial parts and algorithms of this approach are described in the next chapters.

3 Specification of Model Sets with SES/MB

As an appropriate meta-modeling framework the System Entity Structure/Model Base (SES/MB) formalism was determined. This formalism is a general, knowledge based framework. With its key feature to depict the three relationships (i) decomposition, (ii) taxonomy and (iii) coupling it is capable of defining a set of modular, hierarchical models [7] [11] [12]. Decomposition means that the formalism is able to decompose an object into sub-objects. Taxonomy means the ability to represent several, possible variants of an entity. Composition of an entity from sub-entities is done by coupling. This is the meaning of a coupling relationship.

A SES/MB consist of two major parts: (i) a system entity structure (SES) and (ii) a model base (MB). The SES is a tree like structure which contains invariable and variable branches. To create one structure variant the entity structure is pruned, resulting in a pruned entity structure (PES) which is the basis of a composition tree. The composition tree in combination with the model base contains all information to create a hierarchical model.

Figure 3 depicts a SES, represented by a tree structure containing alternative edges starting at decision nodes. With the aid of different edge types and decision nodes a set of different model variants can be defined. To choose a specific design and to create a specific model variant the SES has to be pruned. The pruning process decides at decision nodes which alternative(s) to chose. The result of this process is a Pruned Entity Structure (PES) that defines one model variant.

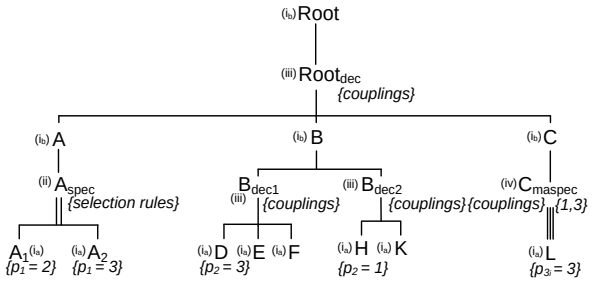

Fig. 3 Node types of a SES tree

The SES formalism differentiates four types of nodes: (i) entity, (ii) specialization, (iii) aspect and (iv) multi-aspect. An entity node represents a system object. There are two subtypes of entity nodes – (i_a) atomic entity and (i_b) composite entity. An atomic entity (figure 3 (i_a)) cannot be broken down into sub-entities. The model base contains a corresponding model for each atomic entity. A composite entity (figure 3 (i_b)) is defined in terms of other entities, which can be of type atomic or composite entity. Thus, the root node of a tree is always of type composite entity, while all leaf nodes are always of type atomic entity. The root node and each composite entity node of the tree has at least one successor node of type - specialization (figure 3 (ii)), aspect (figure 3 (iii)) or multiple-aspect (figure 3 (iv)). The definition of the different node types can be briefly summarized as follows:

- A *specialization node* is indicated by a double line edge. The node defines the taxonomy of a predecessor entity node and specifies how the entity can be categorized into specialized entities. The entity A_{spec} in figure 3 has 2 specializations A_1 and A_2.
- An *aspect node* is indicated by a single line edge. The aspects are variants of decompositions like specializations are kinds of classifications. The decomposition entity B in figure 3 has two aspect nodes B_{dec1} and B_{dec2}.
- A *multiple aspect node* is indicated by a triple line edge. The variable number of its sub-entities is defined by an attached property. The multiple aspect node C_{maspec} in figure 3 defines variants with one, two or three atomic entities L.

A node can have additional properties:

- *Coupling Information* added to an aspect entity – used during the composition of a model structure
- *Attached Variables* added to an entity, e.g. p_1, p_2, p_3 in figure 3 – used for a structure evaluation and as properties for the model
- *Domain Properties* – multiple aspect nodes have attached the possible number of entities.

4 Evolutionary Algorithms

Evolutionary Algorithms (EA) are optimization and search methods that have the biological evolution, mechanics of natural selection and natural genetics, as an archetype. They combine the principle of survival of the fittest among mathematical structures with a randomized information exchange to synthesize a search algorithm. At the beginning of the 60s several research groups modeled the principle of the evolution in mathematical algorithms to develop efficient optimization algorithms: John H. Holland and David E. Goldberg - Genetic Algorithms (GA) [4], Lawrence J. Fogel - Evolutionary Programming (EP) [2], Hans-Paul Schwefel and Ingo Rechenberg - evolution strategy algorithms (ES) [6]. These algorithms are combined under the general term evolutionary algorithms.

The advantages and disadvantages of EAs can be described as follows:

- They model a parallel search in a population of possible solutions.
- They need only marginal knowledge about the problem, e.g. gradient information is not necessary. Hence they can be applied to solve discontinues problems.
- They belong to the group of stochastic search methods. For this reason they can be applied to solve problems where other, traditional methods

e.g. calculus or gradient based approaches are not usable.
- EAs cannot guaranty to find the global optimum.
- EAs consume often a huge amount of computing time and shouldn't be used when a specialized method exists. However because of the parallel search in a population the EAs are parallelizable very well with an almost linear speedup.

In general an EA can be described by the following pseudo code:

```
t = 0
Initialize P(0)
Repeat until stop-criterion is fulfilled
   Evaluate all individual in P(t)
   Select P'(t) from P(t)
   Recombine P'(t)
   Mutate P'(t)
   P(t+1) = P'(t)
   t = t+1
Finish
```

P = population with n individuals

Stop criteria can be:

- Fitness change between P(t-1) and P(t) less than a specific tolerance
- Fitness value below a specific value
- Maximum number of populations, search time, … reached

Another difference to other optimization algorithms is: An EA does not use the parameter set directly, a specific coding to a finite-length string over a finite alphabet is used instead. In the introduced framework the EA has to optimize a parameter set consisting of two parts: (i) model parameters which are a set of numbers and therefore typical parameters for an EA, (ii) structure parameters defined in the SES tree. For the later an algorithm has to transform the tree-coded information to a suitable representation. Both parts together are coded to the finite-length string necessary for the EA. The next section describes among others the tree transformation.

5 A Framework for Modeling, Simulation and Optimization

The fundamental parts of this approach are the interface and method definitions, depicted in figure 4. They bind the established modules: model management, optimization and simulation together to synthesize the framework of a simulation based structure and parameter optimization.

Before an optimization can be carried out, information about the search space, in particular its dimensions, is necessary. In this approach the search space is defined by the set of model structure variants established by analyzing the SES and the set of model parameters, defined by each model structure. During the optimization process several points in the search space are examined. Each point defines one single model structure to be generated through the model generator with one parameter value set.

On closer examination of the framework it is crucial to divide an optimization experiment into two phases:

1. Initialization phase: The model management reads and analyzes a meta-model. Results of the analysis are information about the multidimensional search space (X_S, X_P, D_S, D_P). The optimization module is initialized with this information.
2. Optimization phase: During the optimization phase the optimization method explores the search space within a loop. Each examined search space point, i.e. an ordered set of values (X_{S_i}, X_{P_i}) is delivered to the model management module. This module starts up the processes: structure synthesis, model generation, model simulation and performance estimation. The optimization loop ends when a stop criterion is fulfilled.

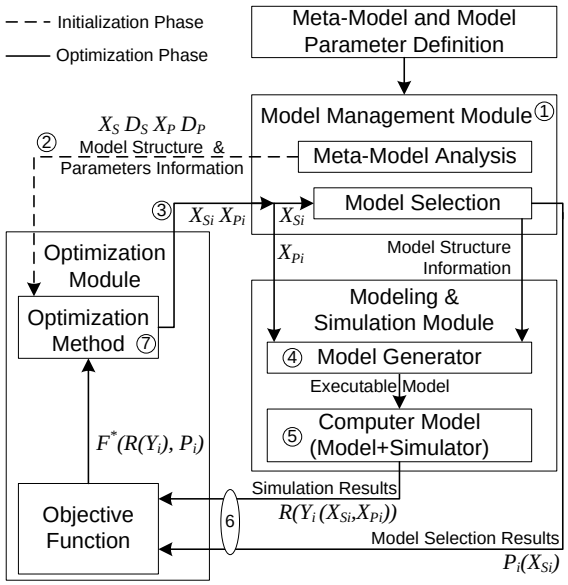

Fig. 4 Structure of the framework

The simulation based optimization framework is segmented into the following modules, methods and interfaces as depicted in figure 4 ((1) … (7)):

1. Model Management Module: meta-model specification: A meta-model definition is read and interpreted by the model management during the initialization phase. The result of this step is a data structure with an SES tree and references to a MB.
2. Interface Model Management Module – Optimization Module (meta-model analysis): In a second step during the initialization phase the model management module analyses the SES tree and establishes the search space. The search space is defined by a set of variables with their domains. These sets X_S, D_S, X_P and D_P are sent to the optimization module.
3. Interface Optimization Module – Model Management Module (transformation of a search

space points into a model configuration): The model management module receives a search space point (X_{S_i},X_{P_i}) within the optimization loop. The sets X_{S_i} and X_{P_i} are used to prune the SES, to synthesize the model structure and to parameterize the model. The selected model structure and model parameters are sent to a model generator as platform and implementation independent XML files.

4. Model Generation Method: Based on the received XML file with model structure information and references to basic components the model generator creates an executable model.
5. Simulation Method: The model is executed by a simulator. In this research the modeling and simulation method is based on the Dynamic Structure DEVS (DSDEVS) formalism [3]. Principally the approach requires a hierarchical modeling method, DEVS is not obligatory.
6. Interface Model Management and Simulator – Objective Function: The objective function gets both simulation results from the simulator and model structure selection results from the model management module to establish the performance of the current model structure and parameter set.
7. Optimization Method: The optimization method establishes the next search space points to examine in a loop until the stop criterion is fulfilled. The search space points are chosen based on the search space definition and chosen optimization algorithm.

5.1 Interface: Optimization Module – Model Management Module

Crucial parts of this research are the interfaces (2) and (3), SES tree analysis during initialization phase and transformation search point + SES → PES during optimization phase.

Within the first interface the Model Management Module has to analyze the SES tree to transform formal meta-model structure information into numerical data useable by the Optimization Module. This is done by a tree analysis starting at the root node, traversing the tree in a defined direction, namely depth-first and breadth-first analysis, and considering every node. If a node is a decision node, i.e. it is a specialization node, multiple aspect node or composite entity node with alternative successor nodes, a structure parameter x_{Si} is added to the structure parameter set X_S and a corresponding domain d_{Si} is added to the domain set D_S. The domains of specialization node and composite entity node are {1, ...,n} with n number of variants. The domain of a multiple aspect node is defined by its attached number range property. Figure 5 illustrates the algorithm for creating structure parameter set X_S and the corresponding domain set D_S based on SES tree information using a breadth-first analysis. The steps and X_S, D_S set build-up order is marked with small sequence numbers.

The breadth-first analysis starts at the root node *A*, a non-decision node. Next nodes are non-decision nodes A_{dec} and *B*. The composite entity node *C* is the first decision node. It has two alternative successors. A first parameter x_{S1} is added to set X_S with the domain d_{S1} *= {1, 2}*. The next examined nodes are B_{dec}, C_{dec1}, C_{dec2}, *D, E, F, G, H* and *I* - they are non-decision nodes. The next node, the multiple aspect node D_{maspec} is a decision node. The value of its number range property is {2, 3, 4}. A second parameter x_{S2} is added to X_S with the domain d_{S2} *= {2, 3, 4}*. The next node, the specialization node E_{spec}, is again a decision node. It has three alternative successor nodes. A third parameter x_{S3} is added to X_S with the domain d_{S3}*= {1, 2, 3}*. The last nodes analyzed *K, E1, E2* and *E3* are non-decision nodes. Thus, the example SES has three decision nodes. The resulting structure parameter set is X_S *= {*x_{S1}*,* x_{S2}*,* x_{S3}*}* with the corresponding domain set D_S *= {*d_{S1}*,* d_{S2}*,* d_{S3}*}* with the above determined domains. On the basis of the combination of these sets X_S, D_S, the model parameter set X_P and its corresponding domain set D_P the optimization method is able to search the search space. Additional SES tree information, e.g. the structure condition at node *A* and the attached variables p_1 and $p_{2i,}$ are irrelevant during the initialization phase.

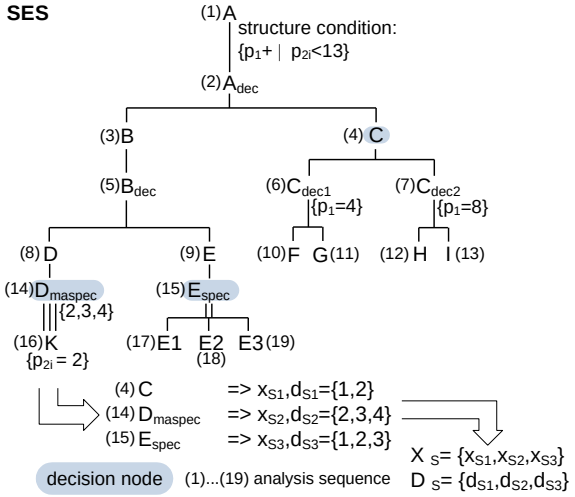

Fig. 5 Transformation SES → sets X_S + D_S

The second transformation is the reverse of the first. The Model Management Module receives a point in the search space from the Optimization Module i.e. the numerical data set $X_i^* = X_{Pi} \cup X_{Si}$, where set X_{Si} codes a specific model structure and set X_{Pi} codes its model parameters. It has to synthesize the corresponding model structure and has to infer the model parameters. The transformation has to traverse the tree in the same direction as during the first transformation in the initialization phase. At each decision node the next element of current structure parameter set X_{Si} is used to decide: (i) which successor of a composite entity node with alternative successors nodes is chosen, (ii) which specialization of a specialization node is chosen or (iii) how many successors of a multiple aspect node are incorporated

into the PES. After pruning, the model structure is verified with the evaluation of all structure conditions. If a structure is invalid, the specific set X_i^* will be refused and this information is sent to the Optimization Module. It marks this point in the search space as prohibited and determines a new one. Figure 6 illustrates the principle of the second transformation. The analysis and pruning order is marked with sequence numbers.

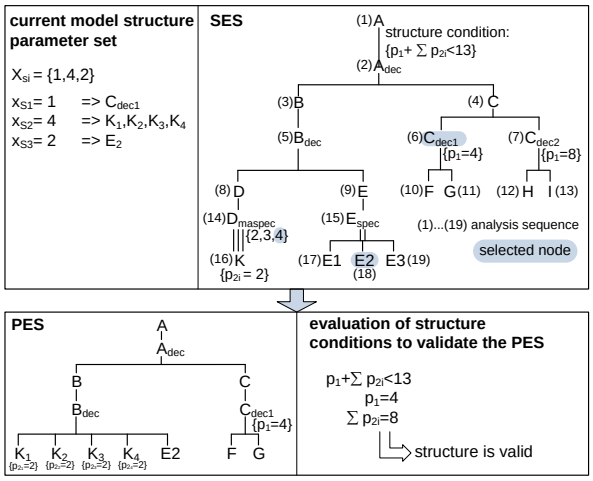

Fig. 6 Transformation X_{Si} + SES → PES

The breadth-first analysis starts at the root node *A* and continues as already described before. The first decision node of the SES tree in figure 6 is composition entity node *C*. The first element in X_{Si} is $x_{S1}=1$, i.e. the first aspect node C_{dec1} is chosen for the PES. The next decision node is the multiple aspect node D_{maspec} and the corresponding set element is $x_{S2}=4$, i.e. the PES contains four nodes *K*. The last decision node is specialization node E_{spec} and the corresponding set element is $x_{S3}=2$, i.e. the PES contains the second specialization *E2* of node E_{spec}. After pruning, the attached variables are calculated and the PES is verified by evaluating the relevant structure conditions. In the example in figure 6, the aspect node C_{dec1}, *E2* and four atomic entity nodes *K* were chosen. Therefore, the structure condition at node *A* is evaluated as follows: $p_1 + \sum p_{2i} = 4 + 8 < 13$. Hence, the PES is valid.

The objective function estimates the performance of the current model configuration. The function has two input sources: (i) simulation results and (ii) information calculated during model generation based on additional attached variables. In figure 6 entities C_{dec1}, C_{dec2} and *D* has attached variables p_1 and p_{2i}, both can be used as additional objective function parameters. After pruning the values of p_1 and p_2 are calculated: $P_{Si} = P(X_{Si}) = \{4; 8\}$.

5.2 Algorithmic Summary of the Framework

As described in the preceding sections, the proposed simulation based parameter and structure optimization framework is composed of different methods that form a uniform optimization approach. The following algorithm, based on the general description in [8], summarizes the fundamental operations using a GA as optimization method.

Initialization Phase:

1. Analyze the SES and establish $X^* = X_P \cup X_S$ and $D^* = D_P \cup D_S$
2. Initialize a population of individuals (generation 0) with different $X_i^* = X_{Pi} \cup X_{Si}$

Optimization Phase (repeat until stop criterion is fulfilled):

1. Estimate the fitness of all individuals of the current generation
   Repeat for each individual
   - 1.1. Prune SES with X_{Si}
   - 1.2. If structure condition is valid, establish $P_i(X_{Si})$ or otherwise mark current individual as invalid and continue with next individual
   - 1.3. Generate model
   - 1.4. Simulate model and get result $Y_i(X_{Si}, X_{Pi})$
   - 1.5. Evaluate the simulation response function $R(Y_i(X_{Si}, X_{Pi}))$ by repeating step 1.4
   - 1.6. Evaluate the objective function $F^*(R(Y_i), P_i)$
2. Select pairs with *m* individuals and create descendants using crossover
3. Mutate the descendants
4. Exchange individuals of the current generation with descendants based on a substitution schema to create a new generation

A disadvantage of a conventional GA is the missing memory. It is possible that in different generations the same individual is repeatedly examined. Because of the time consuming fitness estimation of one individual in simulation based optimization, the addition of a memory method is vitally important. It has to store already examined individuals with their resulting $F^*(R(Y_i), P_i)$. This extension leads to the following, final algorithm summarizing the fundamental operations of the simulation based parameter and structure optimization approach using a GA as optimization method:

Initialization Phase:

1. Analyze the SES and establish $X^* = X_P \cup X_S$ and $D^* = D_P \cup D_S$
2. Initialize a population of individuals (generation 0) with different $X_i^* = X_{P_i} \cup X_{S_i}$

Optimization Phase:

1. Estimate the fitness of all individuals of the current generation
   Repeat for each individual
   - 1.1. Check memory if current individual is known. In case of 'true': continue with

next individual
1.2. Prune SES with X_{Si}
1.3. If structure condition is valid, establish $P_i(X_{Si})$ or otherwise mark current individual as invalid and continue with next individual
1.4. Generate model
1.5. Simulate model and get result $Y_i(X_{Si}, X_{Pi})$
1.6. Evaluate the simulation response function $R(Y_i(X_{Si}, X_{Pi}))$ by repeating step 1.5
1.7. Evaluate the objective function $F^*(R(Y_i), P_i)$
1.8. Store X_i^* and $F^*(R(Y_i), P_i)$ in memory
2. Select pairs with m individuals and create descendants using crossover
3. Mutate the descendants
4. Exchange individuals of the current generation with descendants based on a substitution schema to create a new generation

To prove this algorithm, section 6 describes an industrial application.

6 Parameter and Structure Optimization of a Manufacturing System

To validate the key concept the following application uses developments and problems in the photofinishing industry and investigates a production process. The photofinishing industry specializes in high volume production of thousands to millions of pictures per day but has nevertheless a relatively broad range of different products. As a consequence of significant changes in the photography market, notably the increase of digital orders during recent years, a mix of analogue and digital production facilities are used.

Figure 7 shows general structure and product flow through the different departments of a typical photofinishing laboratory. It is possible to differentiate between three main production departments to depict the production flow analogue film/digital image – photographic picture/end product:

I. The material arrives at the login department. After sorting the product mixes, several single orders are combined into batches. Each batch contains only material for one product type, e.g. specific paper width/surface. The batch creation is done with different machine types: (i) splicers combine undeveloped film rolls onto a film reel, (ii) universal reorder stations (URS) combine analogue reorders to a strap of film strips, (iii) digital URS scan analogue reorders and create digital batches, (iv) digital splicers handle digital data carriers (CDs, flash cards etc.) and (v) software applications combine digital images. Steps (i)/(ii) create analogue and steps (iii)...(v) digital batches.
II. Undeveloped analogue batches have to be developed. Analogue material can be scanned for the next steps which could be digital printing. As an alternative, the analogue batches are printed at analogue printers. The result of both ways is a huge reel of exposed photographic paper.
III. After the development of a photographic paper reel the final step is cutting. Regarding paper cutting both cutter and digital cutter are comparable. A DigiCutter is specialized for paper cutting without a film cutter. Finally, items are packed and identified for delivery to customers.

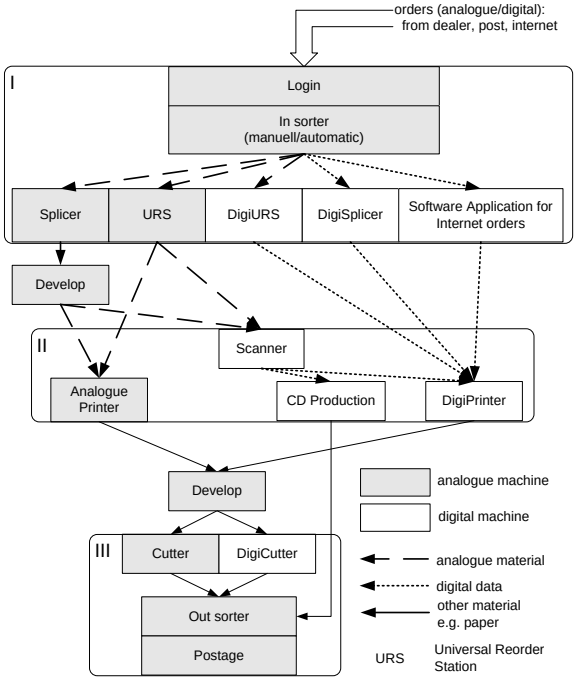

Fig. 7 General product flows of a photofinishing lab

The product flow splicer/URS – analogue printer – cutter was the common production flow before the digital era and is a typical serial manufacturing system. Nowadays there are several possible material routes through production with the same end product but different processing time, machine and operator requirements and costs i.e. a photofinishing lab now appears more as a job scheduling system. It is possible to employ fewer operators than available workstations and produce on time if an appropriate production structure and effective organization method are used to manage production.

For this application the product paths of analogue film material are considered: sorter – splicer – analogue – printer - analogue cutter and sorter – splicer - digital printer - digital cutter. The model has the following structure parameters:

- Automatic and/or manual sorting
- Number of splicer
- Organization strategy login/splicer department
- Organization strategy printer/cutter department

The model has the following model parameter:

- Number of operator in login/splicer department
- Number of operators in printer/cutter department

- Percentage of analogue/digital order handling

Sorting, splicing, printing and cutting of a defined amount of orders takes a production time depending on type and number of machines, number of operators and organization strategies. The production time is estimated by simulation. A production system causes costs. In this case study the costs depends on the number of operators.

For a performance measurement the production of a defined number of orders are simulated. The simulation output of a single run delivers the production time and cost $Y = \{y_{production\ time}, y_{costs}\}$ of the current model variant. The objective function is defined by the term:

$$F = F(Y) = \alpha_1 * r_1 * y_{productiontime} + \alpha_2 * r_2 * y_{costs}$$

The factors α_1 and α_2 normalize the values of the variables, $y_{production\ time}$ and y_{costs}. The factors r_1 and r_2 define the relevance of the variables, $y_{production\ time}$ and y_{costs}. With the factors α_1*=1/max_production_time*, α_2*=1/max_costs*, r_1*=1* and r_2*=1* both variables are within the range between 0 and 1 and have the same relevance. The maximal value of the production time can be calculated with a minimal production system. The maximal value of the costs is defined by the upper bound of the parameter *number of operators ls/pc*.

The challenge for modelling is to minimize the production time and the number of operators.

Figure 8 shows the SES, describing model configurations. It defines 162 model structure variants. Together with the three model parameters, the combination results in 34992 model variants. Not all model variants define useful model configurations, e.g. a model with four operators and eight splicers delivers the same result as a model with four operators and four splicers since in both variants only four splicers can be used. To exclude the useless variants, the root node MODEL defines a structure condition that reduces the number of model variants to 18145.

The search space has to be defined in terms of a structure parameter set, a model parameter set and their corresponding domain sets. Using the principle introduced in section 5 the structure parameter set and the corresponding domain set are defined by:

$X_S = \{x_{DEP\_LOGIN}, x_{controller\_ls\_spec}, x_{splicermaspec}, x_{controller\_pc\_spec}\}$

$D_S = \{d_{DEP\_LOGIN}, d_{controller\_ls\_spec}, d_{splicermaspec}, d_{controller\_pc\_spec}\}$ *with*

$d_{DEP\_LOGIN} = \{1; 2; 3\}$

$d_{controller\_ls\_spec} = \{1; 2; 3\}$

$d_{splicermaspec} = \{1; 2; 3; 4; 5; 6\}$

$d_{controller\_pc\_spec} = \{1; 2; 3\}$

The model parameter and domain set are defined by:

$X_P = \{x_{\#\_of\_operators\_ls}, x_{\#\_of\_operators\_pc}, x_{filter}\}$

$D_P = \{d_{\#\_of\_operators}, d_{\#\_of\_operators\_pc}, d_{filter}\}$ *with*

$d_{\#\_of\_operators\_ls} = \{1; 2; 3; 4; 5; 6\}$

$d_{\#\_of\_operators\_pc} = \{1; 2; 3; 4; 5; 6\}$

$d_{filter} = \{0; 0.2; 0.4; 0.6; 0.8; 1\}.$

To validate the framework the global optimum estimated through simulation of all system variants is compared with the result of an optimization experiment. The simulation results of all variants are shown graphically in figure 9. The complete enumeration estimates 40 global optima with a fitness value 0.26 and 54 local optima with a fitness of 0.27 (error less than 1%). The optimization experiment using the MATLAB GA toolbox has been repeated 100 times. Table 1 shows the results.

Tab. 1 Results of 100 optimization experiments

Average number of investigated individuals to find an global optimum	226,4
Global optimum	47
Near optimal results with max 1% error	26
Results with 1 … 5% error	9
Results with 5 … 10% error	18

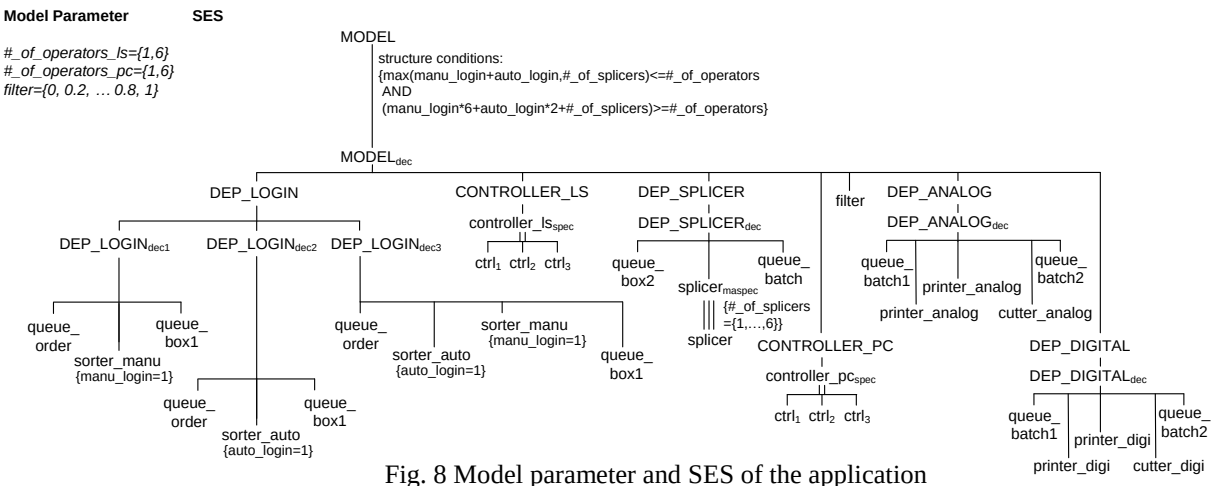

Fig. 8 Model parameter and SES of the application

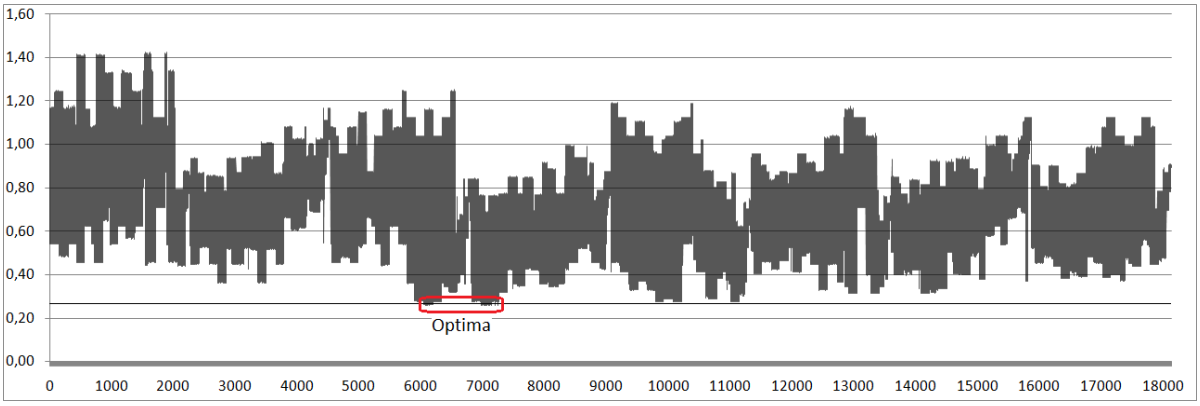

Fig. 9 Fitness values of all variants

The results demonstrate the outstanding advantage of the GA based optimization: on average only 1.2% of the search space has to be examined. It shows also a disadvantage, the not guaranteed finding of the global optimum. But for practical usage the global optimum is not necessary every time, results in the near environment of the global optima are often sufficient.

7 Conclusions and further work

This paper has introduced a simulation based structure optimization method. The approach combines three established methods and extends optimization to the fundamental model structure to enable combined structure and parameter optimization.

It has been shown that using a meta-model as a super ordinate method to define simulation models, parameter optimization can be extended to a combined structure and parameter optimization. Three main elements have been determined: (i) a model generating meta modeling technique based on SES/MB formalism, (ii) a DEVS based modeler and simulator, (iii) an optimization method.

A prototype of the approach was implemented with the Scientific and technical Computing Environment Matlab. The implementation with the MatlabEDSDEVS toolbox [3], MatlabSES [3], implemented within the scope of this research, and Matlab Global Optimization and Parallel Computing Toolboxes by MATHWORKS has been successfully used to prove the approach. The results show that the approach can find an optimal model variant using significant less simulation runs than a complete simulation of all model variants.

[1] F.J. Barros. Modeling and Simulation of Dynamic Structure Discrete Event Systems: A General Systems Theory Approach. PhD thesis, University Coimbra, 1996

[2] L.J. Fogel, A.J. Owens, and M.J. Walsh. Artificial intelligence through simulated evolution. John Wiley, 1996

[3] O. Hagendorf. Simulation Based Parameter and Structure Optimisation of Discrete Event Systems. PhD thesis, Liverpool John Moores University, 2009

[4] J. H. Holland. Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence. The University of Michigan Press, 1975

[5] T. Pawletta, C. Deatcu, S. Pawletta, O. Hagendorf, and G. Colquhoun. DEVS-Based Modeling and Simulation in Scientific and Technical Computing Environments. DEVS/HPC/MMS 2006 Huntsville/Al USA, 2006

[6] I. Rechenberg. Evolutionsstrategie. (German) Friedrich Frommann Verlag, 1972

[7] J.W. Rozenblit, and B.P. Zeigler. Concepts for Knowledge-Based System Design Environments. Proceedings of the 1985 Winter Simulation Conference, 1985

[8] E: Schönberg, F. Heinzmann, and S. Feddersen. Genetic Algorithms and Evolutionary Strategies. (German) Addison-Wesley, 1994

[9] J.R. Swisher, and P.D. Hyden. A Survey of Simulation Optimization Techniques and Procedures. Proceedings of the 2000 Winter Simulation Conference, 2000

[10] A.M. Uhrmacher, and R. Arnold. Distributing and maintaining knowledge: Agents in variable structure environment. 5th Annual Conference on AI, Simulation and Planning of High Autonomy Systems, 1994

[11] B.P. Zeigler, and H. Praehofer, and T. G. Kim. Theory of Modeling and Simulation. Academic Press, 2000

[12] G. Zhang, B.P. Zeigler. The system Entity Structure: Knowledge Representation for Simulation Modeling and Design. Artificial Intelligence, Simulation, and Modeling, Widman L.E., Loparo K.A., Nielsen N.R. (Ed.), John Wiley & Sons Inc, 1989