ON LEARNING ASSISTANCE SYSTEMS FOR NUMERICAL SIMULATION

Irina Bernst, Christof Kaufmann and Jörg Frochte[*].
Dept. of Electrical Engineering and Computer Science
Bochum University of Applied Science
42579 Heiligenhaus, Germany.
[*]*joerg.frochte@hs-bochum.de*

ABSTRACT

The work we present deals with the problem to provide learning assistance systems in the context of simulation and modelling. We develop a classification scheme for learning assistance systems and their use cases. Beyond this, we discuss how learning from simulation data differs from traditional knowledge discovery from data bases. The discussion contains a classification and review of existing approaches followed by an enclosing case study of an assistance system for load balancing purpose in FEM simulation. The presented application case uses a two-stage architecture to minimize additional computational costs. The approach does not require labeled data in the sense of a quality rating for a load distribution nor a teacher for the initial setup and can improve itself unsupervised. For the FEM simulation on heterogeneous distributed systems we introduce a novel feature set and perform an evaluation for several problem sets.

KEYWORDS

Assistance System, Machine Learning, Data Mining, Simulation, Load Balancing, FEM

1. INTRODUCTION

Nowadays numerical simulation in science and product development has become quite standard approach next to classical experiments. With the growing capacity of the computers simulation models grow in size and complexity. Also multi-domain modeling and simulation tools became more and more popular during the last years. To mention a few examples there is COMSOL Multiphysics for FEM simulations or tools based on the modeling language Modelica and collateral relatives like Simscape. The latter class of tools are more related to models which can be represented by differential algebraic equations (DAE). Expressing models with a declarative language increases the freedom in modeling in conjunction with the possibility to make mistakes during the modeling process. A lot of work has already been spent on the question how to assist the user debugging process see e.g. Bunus and Fritzson (2002) or Pop et al. (2012).

As an example for parameter complexity let us consider the dialog *"Model Configuration Parameters"* in Simulink (R2015a) from TheMathWorks. For special use cases e.g. for a real-time HIL simulation, which always comes along with code generation, there are at least three major sub-dialogs that might play an important role: Solver, Optimization and Code Generation. If we just look at the Solver dialog we can choose between many solvers. Especially the fixed-step solvers, like *"ode14x"*, lead to more parameters, such as step size aspects like *"Solver Jacobian method"*, *"Extrapolation order"*, *"Number Newton's iterations"* and so on. So before the simulation can be run, a lot of choices have to be made by the user. This is not the case for a common simulation of a small system on a workstation, since in that case the default settings perform well. But also other tools with code generation capacity like Dymola provide a massive amount of options. Of course

the problem is not limited to real time simulation of DAEs. There are a lot of special demands like e.g. energy or mass conservation that may occur. The selection of proper methods has already been covered for specific applications by some parts of the scientific community, e.g. Stein and Curatolo (1998). Due to the robustness of parameter-independent methods – such as direct solvers for linear systems – these are often preferred over the most efficient and parameter-*dependent* methods – such as domain decomposition and iterative solvers – in many engineering contexts, see e.g. Burrows et al. (2013). Assistance systems could help to choose a method that is superior in a specific application.

In this paper we will deal with the question how to classify learning assistance systems for numerical simulation and modeling. By the term *learning* assistance systems we mean assistance systems, which are able to perform the necessary knowledge discovery and acquisition mainly autonomous, unlike in static approaches as e.g. discussed in Woyand et al. (2012). Therefore we focus on the technique of simulation data mining for building these assistance systems. We use the term simulation data mining in the spirit of Burrows et al. (2011) and similar publications. This combination of simulation and data mining has been successfully applied in different fields such as diagnosis of fluidic systems, see Stein (2003), automotive crash simulation, see Painter et al. (2006), and aircraft engine maintenance, see Mei and Thole (2008). Such a learning approach is able to avoid some common problems like acquiring knowledge by rare experts. Additionally it has the chance to improve itself, which also avoids being limited to an initial knowledge or rule set. Beyond this, being used in a certain application case type, these systems will perform better and better on particular cases of this type, which will be the most interesting cases for the user.

Our contributions in this paper are summarized as follows:

(1) We develop a classification scheme for learning assistance systems and their use cases.
(2) We provide a novel survey how learning from simulation data can be distinguished from traditional knowledge discovery from data bases.
(3) We provide a detailed case study about a machine learning approach for automating the distribution of FEM simulations in heterogeneous environments. To do this we develop and evaluate a working feature set and provide an appropriate performance measure.

The sections of the remainder of this paper is organized along the mentioned contributions starting with section 2 in which we describe the classification scheme. At last the paper is finished with a discussion about the conclusion and future work.

2. CLASSIFICATION SCHEME FOR LEARNING ASSISTANCE SYSTEMS AND USE CASES

This section first describes the common concept of a model hierarchy in modeling and simulation context. Then this hierarchy is used to classify different types of assistance systems, which act on different hierarchy levels.

2.1 Model Hierarchy and Use Cases

Firstly, we present an approach to divide the modeling process into hierarchical steps, which is quite natural and common, see e.g. Finn and Cunningham (1994) or Stein (2008) for related approaches. The purpose of a simulation is to see how a real or mental system reacts to an input or a modification. A user has to build a model which behaves like the system for the observed effects.

So when it comes to simulation the first step is the model. The model starts with the physical behavior of the system and ends with the so-called computer model, which is an executable unit that really runs on a computer system and produces results. It is a long way from the intent to simulate a real or mental system to the computer model. We consider five steps in the modeling procedure:

1. Real or Mental System
2. Physical Model
3. Mathematical Model

4. Numerical Model
5. Computer Model

In our case physical model means the part of the modeling process, where e.g. the engineer mainly works with pen-and-paper. In this first step a lot of decisions are made. Often this is the time for the main simplifications, since only the important effects for the system can be taken into account. E.g. the geometry is simplified or material parameters are assumed to be constant or linear. It is not possible to simulate the real world, but the reaction should be realistic for the considered physical effects. However, there might be several ways or depths to model an effect or even coupled effects. This will make the difference between the real system and the simulated model. If we think about a simple pendulum, the modeler can decide that a mathematical pendulum is sufficient for his purpose. Sometimes these decisions might be performed automatically, because nowadays there exist approaches like Simscape or Modelica Libraries in combination with a proper GUI etc.

When the modeler has fixed his physical model, he needs to find a mathematical expression for it. The mathematical expression is – in contrast to a quite common presumption – not unique. If you think of the pendulum example above one can e.g. describe the equations of motion using a Laplace or a Newton based approach. Considering partial differential equations, the physical model of linear elasticity can be modeled using the Lamé equation or with a saddle point formulation and so on. Making these decisions we end up with the mathematical model. In most cases these decisions will transform the physical model to a mathematical model without further simplifications and assumptions, but in any case these decisions will have implications when it comes to the numerical model.

For a PDE in general there is a wide range of methods like Finite Differences, Finite Elements, Finite Volumes and Spectral Methods, see e.g. Brenner and Scott (2008) and Shen et al. (2011). Some are more or less fixed to a given application domain. Finite Volume methods are for example preferred for hyperbolic PDEs. Let us assume that the initial problem was linear elasticity model and the rough choice was the Finite Element Method (FEM). Whatever one has chosen, the Lamé equation or saddle point formulation, it changes how the FEM is applied on the mathematical model. A saddle point formulation e.g. will require special finite element spaces providing some other benefits etc. Additionally, after one has chosen the discretization in space, one has to choose the discretization in time. Concerning discretization in time one has to decide e.g. between implicit and explicit solvers, Runge-Kutta based approaches or multistep methods and so on. To summarize, the step from the mathematical model to the numerical model requires a lot of decisions.

When all of the numerical methods are selected there is still a very important step left. Most of the numerical methods have parameters to choose or rules of behavioral control. A simple and very common case are fixed step sizes or variable step sizes and rules how to adapt them. The same for non-linear solvers to increase stability, special techniques for higher order DAEs etc.

Nevertheless of course the number of choices a simulation system provide or should provide depends on the scope of the tool. If one considers tools with a very narrow scope e.g. just time-independent linear elasticity models in general all choices except of a very few ones on the lower level of the numerical model are already made. On the opposite we have multi-domain modeling and simulation tools like e.g. Simscape, Comsol or Modelica-based tools, which leave the modeler with a wide range of possibilities in modeling. For these tools the modeler must have deep knowledge to make good choices.

2.2 Hierarchical Classification of Assistance Systems

Now we suggest to classify the assistance systems. They can be classified by the modeling level (see above) in which they assist the user. Hence, like illustrated in Figure 1, we propose to distinguish between a *"Model Assistance System"*, a *"Method Assistance System"* and a *"Parameter and Behavior Assistance System"*.

This classification is independent of whether the assistance system is a learning or a static one. A lot of systems known as expert systems would be considered as assistance system, but they often suffer from the need to acquire their knowledge from human experts and from the problem that they do not grow with the challenges they meet. An update of their abilities is always connected to a software update which again requires a software engineer and an expert in the application area. A learning assistance system avoids these problems. Its ability to improve itself is based on machine learning and pattern recognition techniques like artificial neural

networks or clustering methods. These kind of methods require an initial learning phase that we are going to discuss in section 3.

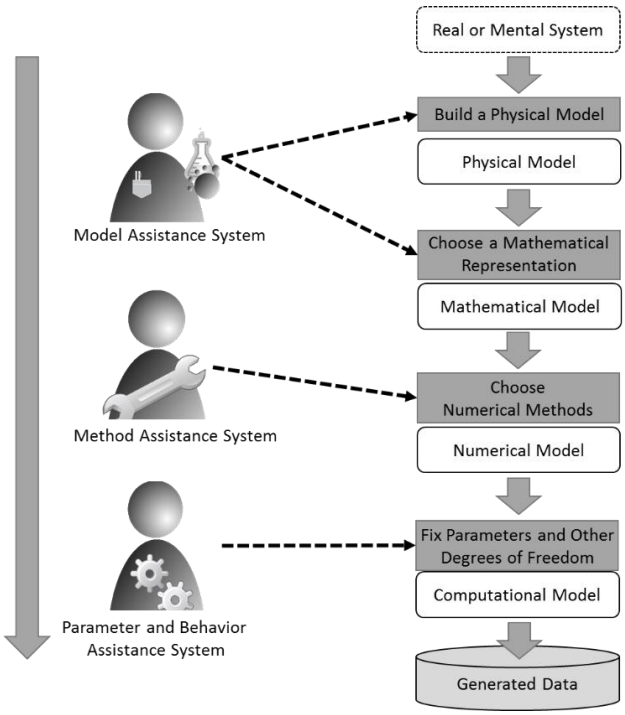

Figure 1. Hierarchical Classification of Assistance Systems

Now we will have a closer look at the different assistance system classes, their characteristics and some related work already done in specific fields. We again restrict us here to learning approaches.

2.2.1 Model Assistance System

The purpose of a model assistance system is to provide help during the beginning of the modeling phase before the actual simulation. On the one hand the more flexible a modeling environment is the more important assistance systems become. But at the same time the development of assistants becomes much harder because the feature selection – which is in general already hard enough – becomes an even more demanding topic. In cases where the range of possible models is limited by the scope of the simulation tool this was already successfully performed by a learning assistance system. For example for the assistance of bridge design processes see e.g. Burrows et al. (2011). Another very interesting application case are problems arising in the context of the diagnosability of technical systems, see e.g. Stein (2003). For very open scenarios the analysis of the model is very demanding. This is due to the problems of analyzing a Modelica model for the purpose of debugging, shown e.g. by Bunus and Fritzson (2002) and Pop et al. (2012). These analyzing techniques are the fundamentals on which one can add the feature selection and processing for assistance systems for these open simulation tool scenarios.

2.2.2 Method Assistance System

Depending on the physical and mathematical model respectively, their discretization and the feature selection, the method assistance system can suggest a numerical method to be used. There has already been done some work in this field concerning learning assistance systems, see e.g. Stein and Curatolo (1998). One has to keep in mind that in general there is no such thing as a *best method* for a given physical or mathematical model. There are some methods that are for an expert or a trained assistance system clearly inappropriate. These can be sorted out. For the rest one needs in general additional requirements from the user. The stable and therefore appropriate methods have different characteristics, e.g. different computational costs, accuracy, conservation of constraints. Depending on the application the user might find e.g. conservation of constraints a very important property or the user has real-time constraints. The latter depends again on the available hardware. An assistance system has to take the preferences of the user and environment information like hardware architecture into account.

2.2.3 Parameter and Behavior Assistance System

This kind of assistance system works on the lower levels, nevertheless it is quite important. The efficiency and stability of most methods depend very much on the chosen parameters or behavior modules. The term *behavior modules* includes aspects like strategies for step size control (see e.g. Cellier and Kofman (2006), section 3.9 and section 4.12) or adaptive refinement of FEM meshes, see e.g. Verfürth (2013). The publication Burrows

et al. (2013) indicates that it is of course possible to design learning assistance systems for FEM models. But as well as Burrows et al. (2011), which focusses on model assistance systems, they both emphasize that the feature selection is very important and non-trivial for spatial models. E.g. FEM output data contains an enormous set of simulation data per mesh node, geometrical data of the mesh itself, material properties and so on. In this context it is very important to form meta-features from this data. In general that cannot be done using automatic approaches. One really has to consider which features make sense and which not. At the end of this consideration the list of candidates might still be too long and one has to make a proper set of features with common approaches. We will discuss that point in our case study – which is based on the publication Bernst et al. (2015) – more detailed.

3. LEARNING FROM SIMULATION DATA – CHALLENGES AND BENEFITS

In this section we will discuss why and how learning from data bases with simulation data differs from ordinary knowledge discovery in databases. According to Fayyad et al (1996) this process contains a few steps that are not necessary when using simulation data. Common databases contain inaccurate or missing data and need a preprocessing step. This is quite different with generated simulation data which is in general *clean*. Such a data base may contain data from failed or timed-out simulations, but this problem is much easier to handle compared to non-generated data. The same holds for the removal of outliers, which is in general not necessary with simulation data.

The quite related task of clustering of similar tasks might be important for simulation data mining, but – in the logic of Fayyad et al. (1996) – for the data mining step and not for the preprocessing step. The reason is that clustering is a – by all means common – approach to group similar cases. Then one can proceed with other techniques like learning a function on a single grouped subset with an artificial neuronal network.

So on the one hand the knowledge discovery in simulation databases for assistance systems seems to be shorter compared to the original one by Fayyad et al. (1996). On the other hand it contains some additional steps that are not possible for an approach on a non-generated data base. Such a data base contains information and there are actions performed like selection, preprocessing or transformation. In general it is quite hard to add a new information feature to an existing data set. This is different for the engineering phase of a learning assistance system with simulated data. There the initial learning phase – which is the more demanding one – is illustrated in Figure 2. This phase can be followed by an online learning approach during the productive phase of the assistance system. First of all let us have a closer look on Figure 2.

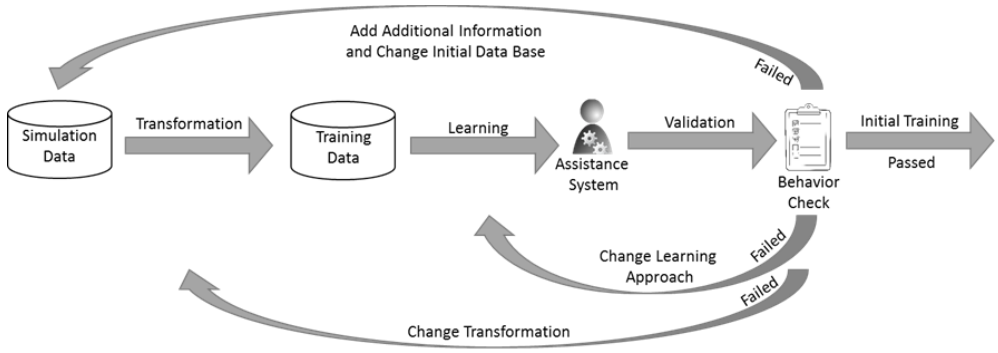

Figure 2. Initial Learning Process

We have some common steps like transformation – which means more or less feature reduction, meta-feature generation and feature selection – and learning. Then the validation might show, if the assistance system works satisfactory. For clustering systems this might mean that different cases are well separated or each sample has enough neighbors. For artificial neural networks (ANN) *satisfactory* might mean that the error is low enough for a validation set. Other validation test cases can be used to check the behavior of the assistance system. If the validation indicates that the system has to be improved, there are three ways shown in Figure 2.

The two lower arrows are accessible for anyone dealing with machine learning or data mining, so also with non-generated data bases. The *"Change Transformation"* arrow is actually a part of the feature selection procedure. The *"Change Learning Approach"* can mean a different method, like ANNs vs clustering based methods, or just a parameter like the number of neurons in an ANN.

However, the top arrow *"Add Additional Information and Change Initial Data Base"* is only possible in simulation data bases. Completely new properties can be recorded. For example if we want to develop an assistance system which is dealing with ordinary differential equations, we might decide after a failed validation in the initial learning phase to e.g. track the changing of the eigenvalues during the simulation. This step can give rise to new features in the training data. Note, this is only possible if one has enough control over the simulation environment, which might be a problem when dealing with closed source software. If we have enough control we are able to generate new features on demand and need. This is a huge difference regarding to a common data base e.g. containing customer data. If this data base does not contain the information one has to work without it. A similar benefit is that we can also generate data on demand. If we are e.g. interested in unusual configuration of a simulation we can perform the simulation and track the results. Therefore we can generate data on demand with different density in the feature space according to our needs. This is again very different to the situation for non-generated data bases. If the data base does not contain enough information e.g. concerning a minority, this situation cannot be changed easily. But nevertheless there is no such thing as a free lunch. Considering ordinary application cases of data mining; the data comes more or less for free. In most cases people work on data that was accumulated not for the purpose of data mining. So for example an internet shop might use the data of its customers for the optimization of the web-shop or the supply of goods. So on the one hand data mining in these data bases is limited to the information given by the costumers or business associates but on the other hand this data comes in principle for free. The benefits of the on-demand generation for training data base of the assistance system is paid by need to spend CPU-time for this purpose. This is especially true for new features, since it might require to start over with the complete data base. So we can now sum-up:

1. In general one will see a two-phase approach for the learning assistance system, first an initial learning phase and then an online learning during the productive phase.
2. For the initial learning phase we have the capability to generate data on demand.
3. Depending on how much control one has about the simulation environment feature generation on demand is possible.
4. The price for top 2 und 3 is computation time and additional effort.

In our case one has to distinguish between top 2 and top 3 because the additional effort for the additional data record is needed just once during the initial learning phase. If we choose to add a new feature, apart from eventually rebuilding the data base in the initial learning phase, and if this feature requires additional effort during the simulation like post-processing, it will in general require the same effort in the productive phase. So to use the example from above: If one considers eigenvalues to be a useful add-on information, one will often try to get along with features that are cheap to compute e.g. Gershgorin circles instead of exact eigenvalues and so on.

So the one challenge for training of an assistance system is to keep the initial learning phase cheap. If all data for the initial phase is just generated for the purpose of training there might show up some scenarios in which the data generation is no more justified by the benefits the assistance system can provide. To negotiate this problem one should use as much data as possible which is generated by skilled users anyway. The more relevant simulation in cloud architectures become, the easier this could be in principle. Users and their simulations on a system without assistance system can be monitored and this data be used for the training of the assistance system. A big issue in this context are data privacy concerns of private people. Also security considerations of companies are still a problem for cloud simulation today, see e.g. Frochte et al. (2014). Much more common are cloud or similar approaches when it comes to education institutes. These tend to use academic licensed products for teaching, see e.g. Bouillon and Frochte (2015). The data acquired here might be helpful if it is mixed up with generated data in the sense of top 2. This will in general be necessary because education institutes might tend to simulate less complex and smaller models than industrial users.

4. CASE STUDY ON AN ASSISTANCE SYSTEM FOR LOAD BALANCING FOR SIMULATION IN HETEROGENEOUS SYSTEMS

In this section we present a case study for an assistance system. It will help in balancing the load from a 3D FEM simulation on a heterogeneous cluster structure. The mesh for the FEM simulation is partitioned using an overlapping Schwarz method. This is a robust domain decomposition method, see e.g. Tosselli and Widlund (2014) or Nikishkov (2007). The major challenge is caused by the heterogeneity of the cluster, since then the optimal distribution is not trivial. Because the problem consists of distributing a computational task, it makes sense to look for two parameters. One is the waiting time, which is associated with convenience of the user and his wish to achieve results quickly. The other is the CPU time, which in a way represents the energy cost as a second optimization variable. This is a very important aspect of cloud computing concerning green IT.

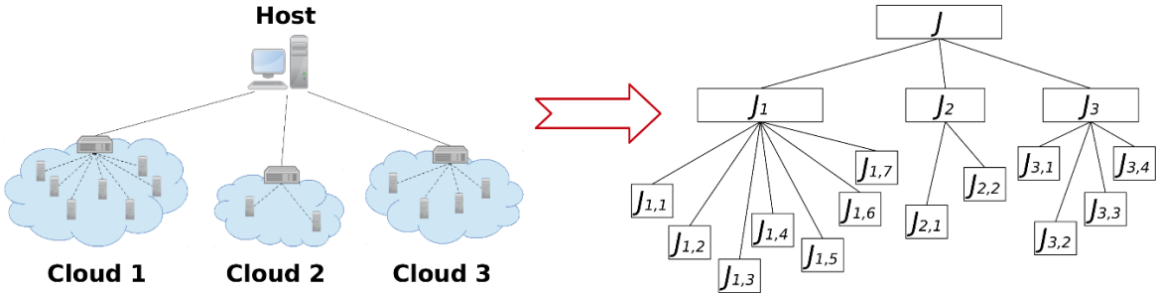

Figure 3. Hierarchical distribution of the job J on a heterogeneous system with three homogeneous sub-systems

We consider a distributed system as shown schematically in Figure 3. The host computer plays an important role in the suggested setting, which triggers the task and manages the computation. It submits the relevant subsets of the simulation data to the remote masters, which again spreads the necessary parts to the computation nodes. The result is a communication cascade, in which one subsystem solves its part of the problem using distributed computing causing network communication between within the cloud. After this is done the remote master of each cloud communicates with host. Because the solving method we use is iterative, this is done multiple times. Most of the communication is done in each subsystem, which makes sense, because the network connection from within a cloud is generally faster than the one between the host and remote masters. Because of the heterogeneity of the system, which means firstly non-equal computation power per computation node across different server-systems and secondly non-equal connection speed between the host computer and the remote masters, an efficient distribution of the tasks is a non-trivial challenge. Because the problem set shown in Figure 3 is of wide scope, we make some reasonable assumptions. We assume that sub-environments are homogeneous among themselves. This means in every sub-environment, e.g. a cloud architecture or a standard blade server, we have the same connection speed between every node, and every node is equal concerning the computation speed.

Load balancing is a technique to optimize the distribution of sub-problems; for a general insight, see e.g. Kameda et al. (1997). Because of the complexity in heterogeneous network environments load balancing is a research field in which heuristic approaches, see e.g. Doulamis et al. (2014), and similar strategies are commonly applied. The learning assistance system in this case study should provide the user with a reasonable load distribution that takes the user's preferences of waiting time and computation time into account. This is similar to the case with the preferences at the time integrator, described in Section 2.2.2. Here the finding of a good distribution depends on the user's preferences as well. In terms of classification defined in Section 2.2.3, this assistance system belongs to the class of *"Parameter and Behavior Assistance System"*.

4.1 Learning Approach and Detailed Task Description

FEM is used for a wide range of problems. Different FEM models like e.g. a model for linear elasticity or one for a laminar flow behaves differently on the same mesh concerning domain decomposition. Problem classes that behave very differently need a different numerical treatment anyway, and thus just problems of the same class can share the same learning data base. The workflow in Figure 4 concentrates as example on linear elasticity problems modeled with the Lamé equation, also known as displacement formulation. There the

process of online learning is illustrated. This assumes an assistance system that has passed already the initial training. This initial learning process is shown in Figure 4.

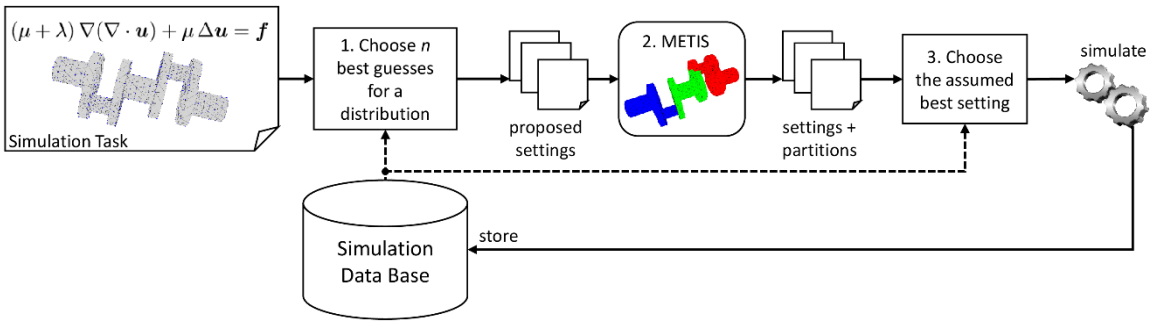

Figure 4. Workflow of the learning method

Very important to understand the used approach is the fact that we have in a way just unlabeled data. Our data base consists of features associated with the waiting time and CPU time of the performed simulation. What our data base does not contain is information about the best choice in a given situation. This is because it is in general a very tricky task to tell whether the setting has been optimal – in most cases there might not be such a thing as a unique optimum. The key idea is to learn the waiting time and the CPU time instead and to use the resulting functions to find good settings.

To achieve this, we argue for a two-stage approach; the first stage uses only a few features, while the second stage uses the full feature set. The reason is that it is quite cheap to evaluate the learned function, but if one needs a partitioning to evaluate the full feature set the call of a partition tool like METIS (Karypis and Kumar (1998)) is not that cheap. To identify the features for the machine learning task in learning workflow we need a more formal description of an exemplary case related to the scenario in Figure 3.

Let J be a computational task (or job) that can be scheduled in parallel. We want to distribute it on a heterogeneous system with a number of major subsystems, like e.g. cloud hardware provider. For example, we consider a system with three subsystems as displayed in Figure 3. Each of them provides a maximal number of computation nodes: c_1, c_2, c_3. Let n_1, n_2, n_3 be the numbers of computation nodes that we use for the computation and which are limited by the maximal number of computation nodes c_1, c_2, c_3.

The job J can be decomposed into maximal three main-jobs J_1, J_2, J_3 (see Figure 3). The size of these jobs concerning the degrees of freedom might differ. These main-jobs can again be decomposed into n_i sub-jobs in the second step up to the number of available computation nodes c_i *(i=1…3)* in this system. The result of this approach is a job hierarchy.

At the beginning of the choice for n_1, n_2, n_3 stands the question, how many computation nodes in total $n_1+n_2+n_3$ the user in general would like to access. This is limited by the number of computation nodes the system provides him with. Because of the job hierarchy it makes sense in the next step to not directly try to answer the question, how we would like to choose n_1, n_2, n_3, but first to answer, how many main-jobs should be in the first level of the job hierarchy and how they should be dimensioned. The last step is how many sub-jobs of each of these main-jobs should be generated and executed. For a more detailed look we will now continue with the feature selection.

4.2 Feature Selection and learning Approach

To start learning our data base must contain information about the hardware, on which the simulation has been carried out. Because gaining data samples is expensive in this application case, it makes sense to restrict us to as few as possible hardware features. As Figure 3 suggests a minimal description consists of

1. number of used computation nodes n_1, n_2, n_3.
2. computation node speed (computation power) in every subsystem s_1, s_2, s_3
3. data transfer rate from J (host) to J_i (e.g. cloud provider): d_{h1}, d_{h2}, d_{h3}
4. data transfer rate in homogeneous sub-environment d_{c1}, d_{c2}, d_{c3}.

Therefore, we end up with 12 features for the hardware topology (HF).

The next step is connected to the first task shown in Figure 3. We have just the model without any partitioning yet. The model consists of the discretized PDE including the right hand side and coefficients, e.g. representing material properties, mesh, the boundary conditions (Dirichlet, Neumann etc.). As mentioned above the suggested trained system is only suitable for a narrow class of problems. Therefore, the PDE itself is not a feature; consider the assistance system trained for this class as a member of a single learning assistance system for a more general problem class. This of course does not cover the right hand side and the coefficients. In Burrows et al. (2013) the aspect of coefficients were discussed. To keep the number of features reasonable we concentrate on the features based on the geometric mesh, which in our case consists of vertices, edges, facets and tetrahedral elements. There are other mesh types for three dimensional problems like hexagonal meshes, but the results and features can easily be transferred to them. Since we cannot use the geometric entities, like vertices, directly as features, we use meta-features as mentioned in Section 2.2.3. Here we use simply:

1. number of vertices in the mesh n_v
2. volume ratio of minimal and maximal volume of tetrahedrons in the mesh, denoted with r
3. number of boundary vertices n_{bv}

The number of vertices n_v as feature correlates to the size of the problem – degrees of freedom – which is the most important aspect. It makes no sense to divide small problems below some limit. The volume ratio r is one of the important aspects, which influence the condition of the problem and its sub problems, see e.g. Brenner and Scott (2008) or Toselli and Widlund (2005). The condition number in turn will influence the behavior of the linear solvers used by the computation nodes. The last feature n_{bv} is related to both. For Dirichlet boundary conditions e.g. for all boundary elements the values are given, so they reduce the degrees of freedom and improve the condition. We denote these selected features as global mesh features (GMF).

This leads to 12 hardware related features and 3 model related features for the first step. Our learning approach proposes to predict the computational cost t_c and the waiting time t_w by a learned function. To do this we suggest multilayer feedforward artificial neural networks (ANN). A design decision in this context is the question whether to use one ANN with two outputs or to use two separate in general smaller ANNs for computation time and waiting time. In our test it turned out that we achieved better results using two ANNs. This might be related to the number of data sets in the training set and may change, if a huge amount of data is available. Our solution can be denoted as follows:

$$t_w = \mathrm{ANN}_1^{waiting}(\mathrm{HF}, \mathrm{GMF})$$
$$t_c = \mathrm{ANN}_1^{computation}(\mathrm{HF}, \mathrm{GMF})$$

By choosing a weight α the user can fix the relation of t_w and t_c for the optimization problem. This is a good example of a user preference that the assistance system has to take into account as mentioned in Section 2. The assistance system cannot know what is more important for the user – cost or time. The goal is to find the minimum of

$$f(n_1, n_2, n_3) = \alpha \frac{t_w}{c_{wait}} + (1 - \alpha) \frac{t_c}{c_{comp}}.$$

This is a problem, which just depends on the chosen number of used nodes, because in our scenario the rest of the features are fixed. In our example with three systems it is a three-dimensional problem. Which technique is used for optimization depends on the size of the search space defined by the total number of used nodes. In many applications it is possible to evaluate all possible combinations and choose the best one; otherwise techniques like the Hill climbing should be used. The scaling coefficients c_{wait} and c_{comp} can be chosen to balance waiting and computation time because in general for distributed problems the computation time is much bigger than the waiting time.

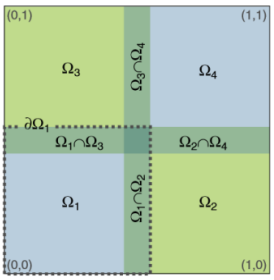

Figure 5. Overlapping Partitions

Using the approach described above we end up with some candidates, this means not just to pick the minimum but a small subset of the best candidates. If we evaluate all variations, we choose the *m* smallest ones; if one uses Hill climbing, one may choose a whole region near the minimum or a union of results of Hill climbing iterations started from different initial points and maybe stopped in local minima.

So, for the next step we partition the mesh once for each of the resulting *m* combinations using METIS. Because of the hierarchical structure of the problem set this is performed in two steps: In the first step three weights, w_1, w_2, w_3 are chosen. These control how many percent of the mesh every partition should consist of before the overlapping part is added. Therefore, these weights represent roughly the sizes of the main-jobs J_i in the first hierarchy level. However, computation is mainly performed on the computation nodes in the second level of the job hierarchy. Hence, the main-jobs are further partitioned into sub-jobs $J_{i,k}$. The corresponding sub-partitions below a main-partition are equally weighted, since all used nodes within one cluster are equal. $w_i = 0$ is interpreted as turn off the specific subsystem. As Figure 4 suggests for this second step, there are additional features available, because now we know more details about how this job will be partitioned.

A new aspect is the size of the overlaps illustrated in Figure 5. This is very important for the convergence speed of the domain decomposition methods. To improve convergence and stability, we let the host *J* and the local master J_i perform a few iterations of a GMRES algorithm, see e.g. Greenbaum (1987), which leads to less network traffic, particularly between *J* and J_i, which is the most expensive one. Nevertheless, on the one hand bigger overlap leads to fewer iterations. On the other hand more parts of the problem are solved redundantly and increase the problem size artificially. Beyond this, we have the effect that concerning the network traffic small overlaps are desirable, because this is the data that has to be transferred in every iteration step. Thus, we have here a non-linear effect that is mainly associated with the number of vertices, which are exchanged between neighbor domains. Therefore, we choose the following feature set per main partition, which means on the first level for all *i* partitions:

1. np_i, number of vertices in the partition
2. rp_i, volume ratio of minimal and maximal volume of tetrahedrons in the partition
3. op_i, ratio of number of overlapping vertices to the partition size.

Because the number of features would increase dramatically, the equally-sized sub-partitions are not considered, and nevertheless their effect is quite deterministic and static using METIS, if one does not optimize the overlapping size as done by Burrows et al (2013) but just keep it fixed as we do it here. If one looks at the features on the second level, one may argue, if concerning some aspects there is less information compared to the first level. One has to keep in mind that n_v correlates with np_i and op_i. A partition equivalent for the global information n_{bv} is missing but for the solution of the sub problems of less importance. With a fixed rule, how to dimension the overlap, the degrees of freedom on the boundaries of the partitions correlate with op_i. For dynamic chosen overlap sizes this information might play a bigger role. We denote the set of all of these features for all main partition as partition mesh features (PMF). Using this together with the hardware features (HF) this leads us to the following approach for the next two ANNs, which are again multilayer feedforward artificial neural networks:

$$\mathrm{t_w} = \mathrm{ANN}_2^{waiting}(\mathrm{HF}, \mathrm{PMF})$$
$$\mathrm{t_c} = \mathrm{ANN}_2^{computation}(\mathrm{HF}, \mathrm{PMF})$$

This network ANN_2 has the potential to give a more accurate answer compared to ANN_1 but at the prize of an increased number of features – in the case of three main-partitions we have 9 PMF instead of 3 GMF. Additionally, each evaluation is preceded by a partitioning procedure to extract the features from the main-partitions. The first aspect leads to the demand of more data for the training and the last makes it more expensive. To emphasize this aspect: this is the main reason for using a two-stage approach. The features for ANN_1 are available just by analyzing the FEM model geometry and the evaluation is cheap. Therefore, it is the first filter selecting a few candidates. For these candidates the partitioning of the main-jobs J_i is performed by METIS, which leads to much higher additional costs but only for these few new candidates. With this

information for these candidates the ANN_2 picks up the most promising distribution. The simulation is then performed, distributed accordingly. When the simulation has finished, we can use the corresponding features with the measured times and add it to our data base to improve the neuronal networks. Because the approach is not based on labeled data, in the sense of knowing the optimal distribution, this leads to a system improving itself in an unsupervised way.

4.3 Performance of the Assistance System

As test problem we choose the linear elasticity model using the Lamé equation with a homogeneous isotropic material parameter and Dirichlet boundary conditions. For training purpose this has been simulated for four three-dimensional objects (geometries), shown in Figure 6, with a mesh size from 250,000 vertices up to 400,000. Because the Lamé equation computes the displacement vector, this means up to 1,200,000 degrees of freedom. The four objects refer to prototypes, like a full convex geometry (sphere), a geometry with a hole (Torus), one with a re-entrant angle (L-shape) and the simple cube.

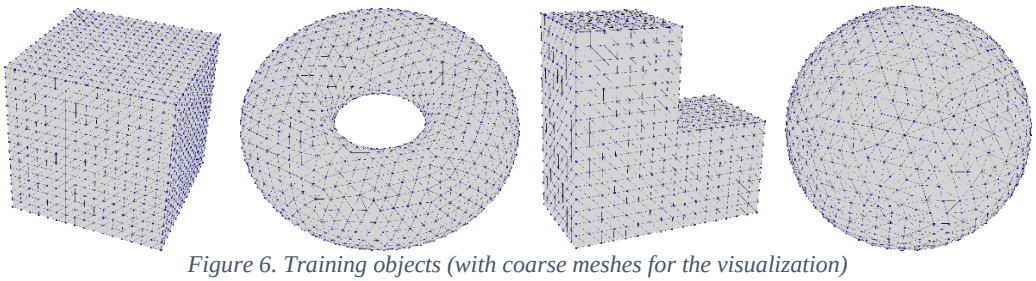

Figure 6. Training objects (with coarse meshes for the visualization)

We provided 564 FEM simulations for different hardware topologies with two and three server systems to prepare the training set. They contain different CPUs, in this case 1) Xeon E5645 @ 2.4GHz, 6-Core, 2) Xeon E5-2660 @2,2GHz 8-Core, 3) Xeon E5-2640 @ 2,5GHz 6-Core. These are constant as well as the connection speed between the nodes in one server system, which differ from server system to server system between fast Ethernet (100 Mbit/s) and Gigabit Ethernet (10 Gbit/s). The connection speed from the host server to each of the server systems has been varied between 10 Mbit/s, 100 Mbit/s and 1 Gbit/s. In this test problem we assume that the user constantly wishes to use 16 computational nodes and at least two server systems, which means clouds in the scenario discussed by Frochte et al. (2014). With the constraint to use always 16 computational nodes we can reduce the number of features by one, because we have $n_3 = 16 - n_1 - n_2$. Furthermore we choose the weights of the main partitions to be $w_i = \frac{n_i}{16}$.

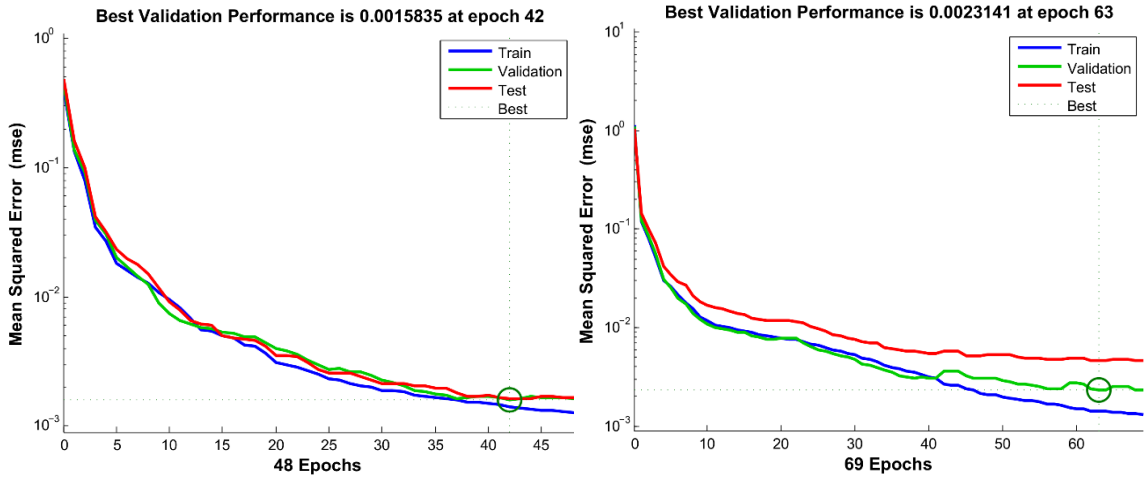

Figure 7. Training of ANN1 (left) and ANN2 (right)

We used the standard approach, see e.g. Marsland (2015), Chapter 2.2, to control the training of the ANNs by dividing our data into three subsets: training, validation and test, as Figure 7 shows.

As mentioned above the ANNs differ concerning their features and as expected they differ concerning their size as well. In our experiments it turned out that the following choices produce reasonable results: ANN_1 for the computation time has 23 neurons and 36 for the waiting time. ANN_2 has for the waiting time 75 neurons and 45 for the computation time. One can sum up that the approximation of the waiting time is more complex than the computation time and that ANN_2 is more complex than ANN_1.

While the difference between ANN_1 and ANN_2 is expected, because ANN_2 has more information to consider, the difference between waiting and computational time might need a further discussion. The communication between host and server system is performed in a network system, which is used unexclusively. This means traffic caused by other users tends to add some additional statistical errors to our measurement.

All the results above, e.g. shown in Figure 7, were achieved on a validation and test set, that is not used for the training. In this case the suggested approach works very well, but the data set just contains geometries, that are used for training. The meshes may differ because of global mesh refinements, but they represent the same geometries with more or less the same ratio between boundaries and inner vertices etc.

For testing the load balancing approach we additionally used three new objects, see Figure 8, which were not included in the initial training, test or validation set above. Ellipsoid is related but not equal to sphere, T-Shape is in a way related to L-Shape and the cube. The hardest geometry is the crankshaft, which is totally different to the training geometries.

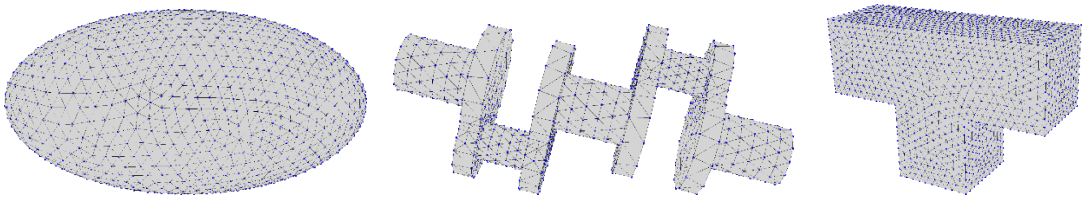

Figure 8. Test objects (with coarse meshes for the visualization)

We tested the quality of predictions of ANNs for foreign objects and geometries on these three. As one can see from Figure 7, the ANN_1 works quite well on our initial data base with known objects. If ANN_1 works so well, does it really make sense to apply a second more expensive step using ANN_2 afterwards? Yes, it makes sense, because ANN_1 loses some of its quality on foreign objects. In our approach the main job of ANN_1 is to work as a filter before using ANN_2.

To do this ANN_1 must mainly keep the relation between the results. If a configuration is better than another in real life, the learned function should keep this relationship. The absolute values are less important. To illustrate effects we picked the shaft as the most demanding geometry. Figure 9 shows the results of 5 randomly picked data base entries with three different values for α. So, the first plot is the use case for optimization of computation time only, the last for waiting time only, and $\alpha = 0.5$ is a mixed scenario. The data is ordered by the value of ANN_1. A theoretical optimal result would be that both graphs are monotonically increasing. Again this figure illustrates that the computation time is easier to predict than the waiting time. For $\alpha = 0$ both are monotonically increasing, in the two other scenarios it is not. So, it turned out in our tests that ANN_1 is able to work as a reasonable filter, but as proposed it is necessary to attach a second step with ANN_2 to increase the probability to pick the best one of the set collected by ANN_1.

The results for the three test objects of Figure 8 are shown in the regression plot in Figure 10. They illustrate that the learned functions correlate with the real world measurements we performed for comparison. Again we see, that it is easier for us to predict the computation time than the waiting time. We assume, that with a bigger data base, the statistical effects concerning the waiting time will be eliminated more and more.

Concerning the waiting time the results suggest that for new objects not always the best solution would be picked up. This corresponds with our tests, in which always a reasonable approach is suggested by the assistance system, but sometimes with expertise knowledge and experience we could provide a better solution.

This assistance system belongs to the class of *"Parameter and Behavior Assistance Systems"*, as already mentioned. This class affects models in the bottom of the hierarchy shown in Figure 1. An assistance system is developed for a specific model type from the class. This also implies that whenever a decision in the modeling process is made differently above in the hierarchy, it will affect this assistance system. If for example in the numerical model the solving method changes, the computation and waiting times might behave completely different. The same holds for example if the mathematical model uses a saddle point formulation instead of the Lamé equation. So for a complete system of assistance systems that support the whole modeling process, there would be a tree hierarchy of assistants necessary.

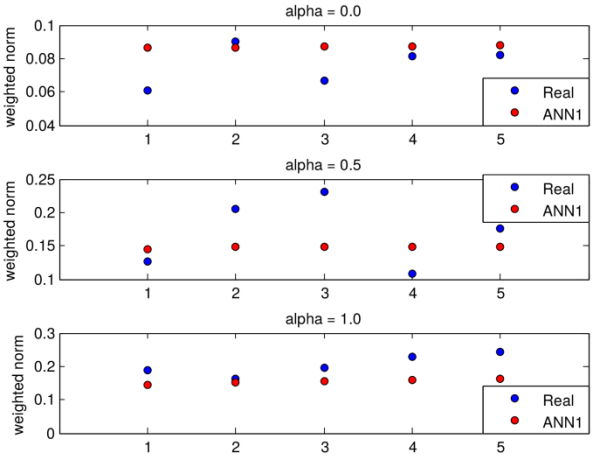

Figure 9. Performance of ANN1 keeping relation the crankshaft geometry

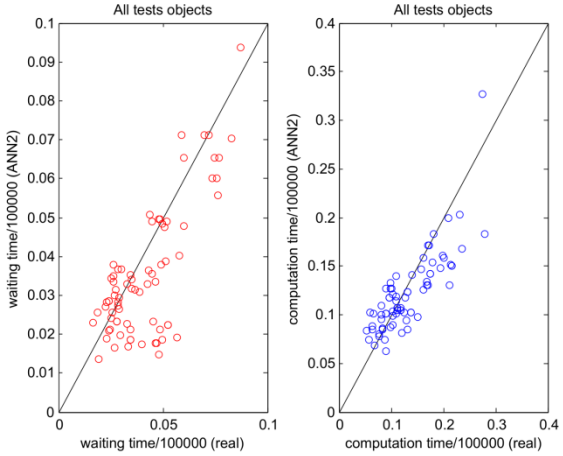

Figure 10. Comparison of ANN2 predictions vs. real on simulation times (scaled) for foreign test objects

5. CONCLUSION

In this paper, we proposed a classification scheme for learning assistance systems and their use cases. The challenge is to provide assistance systems for complex scenarios in simulation for end-users like engineers. We emphasized that static approaches, like e.g. traditional expert systems, cannot keep up with today's demands. For a new generation of self-learning and self-improving systems like e.g. the ones by Burrows et al. (2011) or Stein and Curatolo (1998), we discussed how learning from simulation data differs from traditional knowledge discovery from data bases. The transfer of these concepts to a novel assistance systems on the level of *"Parameter and Behavior Assistance System"* was performed in the presented case study. The goal was an assistance system for load balancing of a simulation task on a heterogeneous hardware architecture. We described the design in detail. To achieve this, our assistance system introduced a novel feature set for a learning approach using artificial neuronal networks. These were implemented in a two-stage design to minimize additional computational costs. The presented work shows how in a complex scenario the decisions of assistance systems on more abstract levels like *"Model Assistance System"* may affect the work of the lower levels. As future prospects we see that modern – especially multi-domain and multi-physics – simulation tools will integrate more and more assistance systems. Sometimes one may realize them because personal preferences will be requested, sometimes they will become the equivalent of assistant systems in automotive and do their work in silence. To achieve this, the interaction of such assistance systems as well as assistance systems particularly on the modeling level need further improvements in future work.

ACKNOWLEDGEMENT

The authors would like to thank Patrick Bouillon for the support & administration of the server systems. This work is supported by the BMBF (Federal Ministry of Education and Research, Germany) under grant 03FH01812. The homepage of the project is located under www.simcloud.eu.

REFERENCES

Bunus, P. and Fritzson, P., 2002. Methods for structural analysis and debugging of Modelica models. In *Proceedings of the 2nd International Modelica Conference*, Vol. 10, pp. 157-165.

Bouillon, P. and Frochte, J., 2015, Simulation- and web-based e-learning in engineering - open source architecture and didactic issues. *In 16th International Conference on Research and Education in Mechatronics*, pp. 127-134, IEEE.

Bernst, I., Kaufmann, C. and Frochte, J., 2015. Learning Load Balancing for Simulation in Heterogeneous Systems. *Proceedings of the 12th Intern. Conference Applied Computing*, Maynooth, Ireland, pp. 121-128.

Brenner, S. and Scott, L., 2008. *The mathematical theory of finite element method, 3rd ed.* Springer-Verlag, New York-Berlin-Heidelberg.

Burrows, S., Stein, B., Frochte, J., Wiesner, D. and Müller, K., 2011. Simulation Data Mining for Supporting Bridge Design. *Proceedings of the Ninth Australasian Data Mining Conference.* Australia, pp. 163-170.

Burrows, S., Frochte, J., Völske, M., Martínez Torres, A.B. and Stein, B., 2013. Learning Overlap Optimization for Domain Decomposition Methods. *Proceedings of the 17th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 13)*, pp. 438–449.

Cellier, F.E. and Kofman, E., 2006. *Continuous System Simulation*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Doulamis, N., Kokkinos, P.C. and Varvarigos, E.A., 2014, Resource Selection for Tasks with Time Requirements Using Spectral Clustering, *IEEE Trans. on Computers*, Vol. 63, No.2

Fayyad, U., Piatetsky-Shapiro, G. and Smyth, P., 1996. From data mining to knowledge discovery: an overview. In *Advances in knowledge discovery and data mining*. American Association for Artificial Intelligence, Menlo Park, CA, USA, pp. 37-54.

Finn, D. P. and Cunningham, P., 1994. Physical Model Generation in Thermal Engineering Problems described by Partial Differential Equations. In *Proceedings of the Eighth International Workshop on Qualitative Reasoning about Physical Systems* (QR 94), pp. 90-97.

Frochte, J., Kaufmann, C. and Bouillon, P., 2014. An approach for secure cloud computing for FEM simulation. *Proceedings of the 11th International Conference Applied Computing*. Porto, Portugal, pp. 234–238.

Greenbaum, A., 1997. *Iterative Methods for Solving Linear Systems.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

Kameda, H, Li, J., Kim, C. and Zhang, Y., 1997. *Optimal Load Balancing in Distributed Computer Systems*. Springer-Verlag, London, UK.

Karypis, G. and Kumar, V., 1998. A fast and high quality multilevel scheme for partitioning irregular graphs. *Journal on Scientific Computing.* Vol. 20, No. 1, pp. 359–392.

Marsland, S., 2015. *Machine learning - An Algorithmic Perspective*. CRC Press a Chapman& Hall book, Boca Raton.

Mei, L. and Thole, C. A., 2008. Data analysis for parallel car-crash simulation results and model optimization. *Simulation modelling practice and theory*, Vol. 16, 3, pp. 329-337.

Nikishkov, G.P., 2007. *Basics of the domain decomposition method for finite element analysis*. Saxe-Coburg Publications, Kippen, UK.

Painter, M. K., Erraguntla, M., Hogg, G.L., and Beachkofski, B., 2006. Using simulation, data mining, and knowledge discovery techniques for optimized aircraft engine fleet management. In *Proceedings of the 38th conference on Winter simulation, pp. 1253-1260. Winter Simulation Conference.*

Pop, A., Sjölund, M., Asghar, A., Fritzson, P. and Casella, F., 2012. Static and dynamic debugging of Modelica models. In *Proceedings of the 9th International Modelica Conference (Modelica'2012)*, Munich, Germany.

Shen, J., Tang, T., and Wang, L.-L., 2011. *Spectral Methods: Algorithms, Analysis and Applications* (1st ed.). Springer Publishing Company, Incorporated.

Stein, B., 2003. Model compilation and diagnosability of technical systems. In *Proceedings of the 3rd International Conference on Artificial Intelligence and Applications (AIA 03)*, Benalmádena, Spain, pp. 191-197.

Stein, B., 2008. Model construction for knowledge-intensive engineering tasks. In *Advances of Computational Intelligence in Industrial Systems.* Springer Berlin Heidelberg, pp. 139-167.

Stein, B. and Curatolo, D., 1998. Selection of Numerical Methods in Specific Simulation Applications. *Proceedings of the Eleventh International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems.* Castellon, Spain, pp. 918-927.

Toselli, A. and Widlund, O., 2005. *Domain Decomposition Methods – Algorithms and Theory, 1st ed.* Springer-Verlag, Leipzig, Germany.

Verfürth, R., 2013. *A Posteriori Error Estimation Techniques for Finite Element Methods*. Oxford University Press.

Woyand, H., Goldhammer, L. and Roj, R., 2012. A knowledge based assistance system for finite element analysis. In *15th International Conference on Interactive Collaborative Learning (ICL),* pp. 1-4.