Knowledge Discovery for Pareto based Multiobjective Optimization in Simulation

Patrick Lange, Rene Weller, Gabriel Zachmann
University of Bremen, Germany
{lange,weller,zach}@cs.uni-bremen.de

ABSTRACT

We present a novel knowledge discovery approach for automatic feasible design space approximation and parameter optimization in arbitrary multiobjective blackbox simulations. Our approach does not need any supervision of simulation experts. Usually simulation experts conduct simulation experiments for a predetermined system specification by manually reducing the complexity and number of simulation runs by varying input parameters through educated assumptions and according to prior defined goals. This leads to a error-prone trial-and-error approach for determining suitable parameters for successful simulations. In contrast, our approach autonomously discovers unknown relationships in model behavior and approximates the feasible design space. Furthermore, we show how Pareto gradient information can be obtained from this design space approximation for state-of-the-art optimization algorithms. Our approach gains its efficiency from a novel spline-based sampling of the parameter space in combination within novel forest-based simulation dataflow analysis. We have applied our new method to several artificial and real-world scenarios and the results show that our approach is able to discover relationships between parameters and simulation goals. Additionally, the computed multiobjective solutions are close to the Pareto front.

CCS Concepts

•Computing methodologies → Model development and analysis;

Keywords

Knowledge Discovery in Simulation; Multiobjective Optimization; Association Rule Mining; Spline Interpolation; Correlation Analysis; Pareto Principle

1. INTRODUCTION

Traditional simulation-based optimization approaches [25, 21] usually require pre-defined objective functions in order to use optimization techniques to find a local or global minimum for the specified simulation. Unfortunately, such objective functions are not available in blackbox simulation problems. In these blackbox simulations, but also in regular simulations, is the analysis of the model behavior and the determination of the valid design space, respectively, usually done manually by simulation experts. This manual analysis is generally performed by identifying a few distinct parameters according to the simulation project scope given as a set of simulation goals. An optimization is not conducted by a simulation itself, but rather through execution of multiple simulation runs. In order to reduce complexity and number of runs, the input parameters of each single run have to be varied cleverly [27]. The simulation expert usually takes an educated guess based on his experience which parameters might be influential on the project scope and therefore time and effort is invested in experimenting with these focus parameters in a fixed system configuration environment. Hence, for each simulation run, the input configuration has to be adjusted if the simulated model performance does not meet scenario or engineering expectations. Currently, this adjustment of input parameters in simulations is either done externally by simulation experts that need to guide the simulation process, or by defining a number of scenarios. These scenarios are pre-defined by simulation experts to cover almost all aspects of the simulation model. The use of expert guidance can lead to quite effective simulation results. However, such experts are rare and expensive. Additionally, its not always feasible to have an expert available for configuring and supervising the simulation. Nevertheless, this approach is widely used [13] but yields many disadvantages as this workflow is based upon subjective judgment of simulation results. These judgments are insufficient for efficiently solving this problem because they can not survey the whole underlying multiobjective problem of the simulation model, especially for blackbox simulations.

[13] refers to this as the "trial-and-error approach" to finding a good solution and recommends that simulation experts should spend more time in analyzing than building the model. Furthermore, pre-defined scenarios may lead to less optimal adaptation [12].

Consequently, it would be beneficial to automatically compute suitable input configurations for a given simulation model without the need of an expert guiding this process [28]. In addition, recent simulation models are dominated by a multiobjective optimization problem (MOP) because many real world problems involve decisions based on multiple and conflicting criteria [17].

SIGSIM-PADS '16, May 15 - 18, 2016, Banff, AB, Canada

ISBN 978-1-4503-3742-7/16/05…$15.00

DOI: http://dx.doi.org/10.1145/2901378.2901380

There is already a number of computational methods for solving MOP [9, 32] available. However, they usually do not consider the generation of vast amounts of simulation model behavior results that can be easily derived from simulation data farming. Although, this data could be used to deliver additional (gradient) information to traditional MOP solving approaches. In the best case, optimization approaches can directly benefit from this information.

In this paper, we present a different approach which is directly based on this observation and the idea of so-called *Knowledge Discovery in Databases (KDD)* [36]. Unlike traditional approaches for solving MOP, KDD in simulations is not limited to a static, pre-determined input dataset for model behavior and optimization. Instead, the simulation can be used as an generator for new data by itself. This enables us to investigate the whole bandwidth or at least the largest part of possible model behavior by conducting cleverly designed simulation data farming in order to discover surprises and potential [11, 35] which can be re-used in the MOP solving process, too. More precisely, we adapt techniques from KDD research to multiobjective Pareto optimization in blackbox simulations with unknown objective functions. Our approach autonomously builds an active model between simulation input and simulation goals which is capable of

- approximating the feasible design space for a Pareto based optimization by uncovering unknown causal relations in large parameter sets between simulation input and model behavior which are assumed to be unknown non-linear objective functions.
- computing Pareto gradient information from this design space approximation which can be used in state-of-the-art multiobjective optimization solvers.

As our approach is completely autonomous, it does not need any supervision from simulation experts. Another advantage of our approach is its performance. It gains its efficiency from a novel spline-based sampling of the parameter space in combination with a novel forest-based simulation dataflow analysis. The only pre-condition is that the simulation output has to be deterministic.

2. RELATED WORK

Research in combining KDD and simulation methodology has attracted increasing interest in the last decade. [24] explored the landscape characterization problem with a support vector machine (SVM) by analyzing the complete input parameter space. The approach assumed a non-goal oriented simulation, in which the simulation model can be reduced to a single function f which updates the simulation state x with parameter set θ via $x_{k+1} = f(x_k, \theta)$. They defined the landscape characterization problem by determining the set of points θ in which a pre-defined simulation state is achieved. This approach can neither be applied to single-objective nor multi-objective based simulations in which the simulation model is governed by a set of (possible) contradictory functions $f_i, ..., f_n$ as the approach does not concern any contradictory goals within the simulation. [4] determined dynamic adaptation strategies for agent-based traffic simulations via supervised learning. They extracted parameter patterns in the from of decision trees in stochastic simulation by simulating the simulation model several times. These generated decision trees are valid for linear relationships between input parameters and model "what if" studies. The approach is further restricted to a small number of simulation input parameters as the approach involves a runtime which is quadratic in the number of input parameters. Likewise, [26] neglected multiobjective simulation properties. They investigated the application of KDD in simulation of aircraft engine fleet management. They applied a linear regression to all input parameters $x_i, ...x_n$ for one simulation goal state y resulting in a model of the form $y = C + \alpha_1 x_1, ..., \alpha_n x_n$. This model was used to determine the cost drivers in aircraft fleet management. These cost drivers were then classified by a clustering algorithm into low- or high cost classes, describing the main cost drivers for the given fleet management simulation. [27] proposed an approach for uncovering unknown relationships in model behavior. They conducted large scale experiments by replicating pre-defined experiment definitions. The resulting simulation data output was clustered and presented in various plots and charts in order to reveal unknown relationships for the simulation experts and is consequently highly depending on the simulation expert. KDD approaches based on multiobjective (respectively Pareto) simulations (such as [34, 20, 23, 6]) focussed on extracting additional information from pre-determined Pareto sets or analzying these sets within the simulation. Consequently, they can be used to neither approximate the feasibile design space nor to compute a Pareto solution itself.

In summary, all of the above mentioned studies were focused on building passive models between simulation input and goal-related simulation output while minimizing the simulation parameter scope or by focusing on single-objective linear simulation models. These passive models deliver coarse granularity parameter relationship information which can be used to neither approximate the feasible design space nor to compute a Pareto gradient information (e.g. distance or gradient information of the analyzed data with respect to the Pareto front). Moreover, they are highly depending on the simulation expert supervising the KDD process. In addition, they are also not applicable to non-linear simulation models in which the objective functions are not available. Consequently, they can not be used as input for multiobjective optimization algorithms in order to compute suitable configurations.

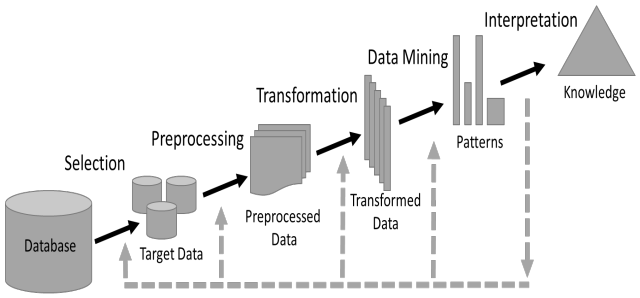

Figure 1: The traditional Knowledge Discovery in Databases (KDD) process: Low level data is extracted and data mining methods generate specific representations. Manual evaluation of these representations leads finally to low level data knowledge. Adapted from [36].

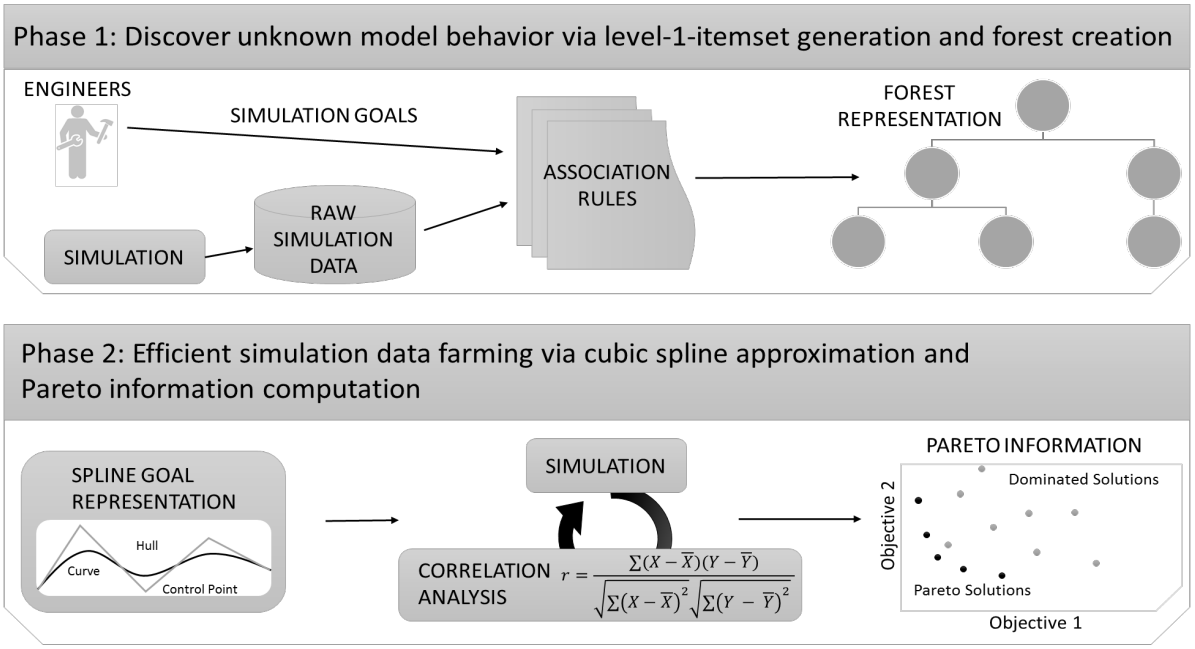

Figure 2: Our autonomous knowledge discovery process: First, all causal relations between input parameters and simulation goals are uncovered. Second, simulation data farming is efficiently conducted in order to approximate the feasible design space as well as to compute Pareto gradient information.

3. OUR KNOWLEDGE DISCOVERY PROCESS

Originally, Knowledge Discovery in Databases (KDD) is defined as making sense of data collections that are too big to manually review each and every single record. Input sources for such kinds of data are complex simulations, graphs, or data warehouses [37]. [36] describe the KDD process as multiple steps to ultimately transform low level data into useful knowledge (see Figure 1). In detail, the KDD process is a highly interactive five-step-process that requires many decisions made by the user. Some of these steps (e.g. target data selection or interpretation of patterns) have to be iteratively repeated by the user for convincing results. Hence, KDD is a semi-automatic process because the user is ultimately responsible for interpretation and evaluation of mining results. This particularly applies for the evaluation of the usefulness of the generated knowledge [36].

Today, simulation models are dominated by a multiobjective optimization problem (MOP) because many real world problems involve decisions based on multiple and conflicting criteria [17]. The optimal decisions have to consider the best trade-off among these criteria. This is actually the goal of multiobjective optimization. Such multiobjective optimization problems can be found in many situations, for example, in product design where several criteria must be simultaneously satisfied [5, 31, 33]. We define MOP according to [17]: Given a subset X of $\mathbb{R}^n$ and p functions $f_j : X \Rightarrow \mathbb{R}$ for $j = 1, 2, ..., p$, MOP is defined as:

$$(MOP) \min_{x \in X} F(x) = (f_1(x), f_2(x), ...f_p(x)) \qquad (1)$$

where $F : X \Rightarrow \mathbb{R}^p$ is the objective function vector. We assume that X is of the form $X = \{x = (x_1, x_2, ..., x_n) \in \mathbb{R}^n : a_i \leq x_i \leq b_i, i = 1, 2, ..., n\}$, where a_i and b_i are the lower and upper bound of the ith component of variable x, respectively. When the objective functions conflict with each other, no single solution can simultaneously minimize all scalar objective functions $f_j(x), j = 1, ..., p$. Consequently, it is necessary to introduce a new notion of optimality in multiobjective problems. A most commonly used one is that of Pareto optimality or Pareto efficiency, which is an important criterion for evaluating economic and engineering systems. The definition of Pareto optimality can be provided by using Pareto dominance relation [1]:

- Let $x_u, x_v \in X$ be two decision vectors. $F(x_u)$ is said to dominate $F(x_v)$ (denoted $F(x_u) \prec F(x_v)$) if and only if $f_i(x_u) \leq f_i(x_v)\ \forall i \in \{1, 2, ..., p\}$ and $f_j(x_u) < f_j(x_v)\ \exists j \in \{1, 2, ..., p\}$
- A point $x* \in X$ is globally Pareto optimal if and only if there is no $x \in X$ such that $F(x) \prec F(x*)$. Then, $F(x*)$ is called globally efficient. The image of the set of globally efficient points is called the Pareto front. In general, computational methods cannot guarantee global Pareto optimality [18], but at best local Pareto optimality that is defined as:
- A point $x* \in X$ is locally Pareto optimal if and only if there exists an open neighborhood of $x*, B(X*)$, such that there is no $x \in B(x*) \cap X$ satisfying $F(x) \prec F(x*)$. $F(x*)$ is then called locally efficient. The image of the set of locally efficient points is called the local Pareto front.

The goal of MOP is to identify a subset of the Pareto optimal points ($\mathcal{P}*$) which is able to represent the Pareto front or to compute a single trade-off solution $x \in \mathcal{P}*$. In general, identifying the set of all Pareto optimality points is not a tractable problem and mostly impossible, particularly when the knowledge on the structure of the problem is very minimal or not available [17].

In this work, we present the application of an completely automatic knowledge discovery process to reveal causal relationships between simulation input parameters and predefined simulation goals with respect to blackbox simulations with an underlying multiobjective model behavior. The result of our knowledge discovery process is an approximation of the feasible design space as well as Pareto information which can be directly used for solving the multiobjective optimization problem of the model.

Three main challenges arise when applying KDD techniques to these problems:

First, engineers who specify the simulation model as well as simulation experts have limited and hence incomplete knowledge about the simulation model behavior with respect to the complete parameter input space. Consequently, the assumed relations between pre-defined simulation goals and parameter space input are incomplete or wrong. Unfortunately, efficiently computing viable solutions for MOP requires at least a correct approximation of the relationship between the parameter input space and the objective functions. This means, all relations between a simulation input parameter and a pre-defined simulation goal within the simulated model behavior have to be determined. Second, KDD requires extensive simulation data farming in order to yield useful results. This simulation data farming can lead to a computationally very expensive KDD process because this complexity usually grows at least quadratically with the amount of input parameters [4]. Hence, the simulation data farming constraints (selection and sampling of convenient input parameters) have to be minimized. Third, diverse algorithms exist for computing a solution to MOP, such as gradient descent [15, 14], simulated annealing [8, 32] or evolutionary algorithms [9, 10]. The proposed knowledge discovery process should yield Pareto information in such a way that this information can be directly used in such different optimization approaches.

In order to overcome these challenges, our knowledge discovery process differs in many ways from the above described standard KDD process (see Figure 2). Basically, our process is split into two main phases: association rule based dataflow analysis and simulation data farming with relationship analysis. The first phase reveals unknown model behavior and constructs a simulation goal based forest data structure which enables fast simulation data farming. The second phase utilizes this forest in order to analyse the unknown model behavior. This analysis is used to approximate the feasible design space and consequently to compute the needed Pareto information for MOP optimization. In the following, we will detail both phases of our main algorithms.

4. FOREST-BASED ASSOCIATION RULE MINING

Our knowledge discovery process starts with the determination of possible causal relations between simulation input parameters and simulation goals. We define a possible causal relation with an existing dataflow inside the simulation denoted as $f_j\{x_i, ..., x_n\} \mapsto \mathcal{G}_j$ where $\mathcal{G}_j$ is a pre-defined simulation goal which maps the parameters $\{x_i, ..., x_n\}$ with an objective function f_j to a satisfaction value or goal state. Since our approach assumes a blackbox simulation, no mapping between parameters $\{x_0, ..., x_n\}$ and simulation goals $\{\mathcal{G}_0, ..., \mathcal{G}_j\}$ as well as explicit forms of $\{f_o, ..., f_j\}$ is known in advance. The simplest, and computationally most expensive, approach would be to brute-force analyze all given parameters for every simulation goal in order to reveal unknown model behavior. This would result in a simulation data farming computational complexity of:

$$\mathcal{O}((n^2 - n) \cdot g) \tag{2}$$

where

g : number of simulation goals
n : number of simulation model input parameters

Sophisticated simulations easily inherit hundreds or thousands of input parameters with large parameter spaces. This would result in computationally very expensive brute-force analysis of the complete knowledge discovery process.

In order to overcome this limitation, we present several ideas to accelerate the computation. We start with a fast Association Rule Mining (ARM) which uncovers the complete dataflow of the simulation by analyzing all dataflow transactions. These transactions can be used to determine the parameter mapping $\{x_i, ..., x_n\}$ as well as for identifying f_j of $f_j\{x_i, ..., x_n\} \mapsto \mathcal{G}_j$.

The main idea is to use the traditional ARM Apriori [30] algorithm with low support and high confidence settings. In order to enable data farming parallelization and pruning of the analyzed workflow, we transform the original list output of the Apriori algorithm into a disjoint union of tree data structures called forest.

Following the original definition by [30], the problem of ARM is defined as: Let $I = i_i, i_2, ..., i_n$ be a set of n binary attributes called *items*. Let $D = t_1, t_2, ..., t_n$ be a set of transactions called the database. Each transaction in D has a unique transaction ID and contains a subset of the items in I. An association rule is an implication expression of the form $X \Rightarrow Y$, where X and Y are disjoint itemsets, i.e., $X \cap Y = 0$. Further, $X, Y \subseteq I$.

To illustrate these concepts, we use a small example from the supermarket domain: $\{butter, bread\} \Rightarrow \{milk\}$ meaning that if butter and bread are bought, customers also buy milk.

The strength of an association rule can be measured in terms of its support and confidence. The support value of X with respect to T is defined as the proportion of transactions in the database which contains the item-set X given as $\sigma(X) =\mid \{t_i \mid X \subseteq t_i, t_i \in T\} \mid$. Confidence, on the other hand, measures the reliability of the inference made by a rule. Both are mathematically defined as:

$$Support, s(X \Rightarrow Y) = \frac{\sigma(X \cup Y)}{N} \tag{3}$$

$$Confidence, c(X \Rightarrow Y) = \frac{\sigma(X \cup Y)}{\sigma(X)} \tag{4}$$

Apriori [30] is an algorithm for frequent item set mining and association rule learning over transactional databases. It proceeds by identifying the frequent individual items in the database and extending them to larger and larger item sets as long as those item sets appear sufficiently often in the database. The output of the Apriori algorithm is a list of level-k-itemsets: $\{\{X_0, ..., X_k\} \Rightarrow Y\}_{s,c}$.

Algorithm 1 GenerateForestStructure

$\mathcal{O}$ = list of objective references $\{X_0, ..., X_g\}$
$\mathcal{L}$ = list of level-1-itemsets rules: $\{\{X\} \Rightarrow Y\}$ from the Apriori algorithm
$\mathcal{F}$ = forest root node
for $\mathcal{O}_i \in \mathcal{O}$ **do**
  $\mathcal{M}$ = tree root node with $\mathcal{O}_i$
  $\mathcal{M}_{childs}$ = GenerateTreeStructure($\mathcal{M}$, $\mathcal{L}$, $\mathcal{O}_i$)
  $\mathcal{F}_{trees}$ += $\mathcal{M}$
end for

Algorithm 2 GenerateTreeStructure

$\mathcal{R}$: read relations of $\mathcal{O}_i$ as $\bigcup_{X \Rightarrow Y \in \mathcal{L}} := \{X \mid \forall Y = \mathcal{O}_i\}$
for $\mathcal{R}_i \in \mathcal{R}$ **do**
  $\mathcal{C}$ = child node of $\mathcal{M}$ with $\mathcal{R}_i$
  $\mathcal{C}_{childs}$ = GenerateTreeStructure($\mathcal{C}$, $\mathcal{L}$, $\mathcal{R}_i$)
end for
return $\mathcal{M}$

In our scenario, we are only interesting in direct relations represented as consistent association rules. More precisely, we are interested in level-1-itemset rules which have high confidence c and low support s as they describe direct parameter relations [30]. Due to the inherent structure of a simulation dataflow, which is constituted by the simulation workflow, repeating patterns of data access emerge. For instance, a physically-based simulation of Newton's law will always modify the position and velocity of certain simulated objects. This physically-based simulation will update the corresponding objects every time step in the simulation, generating such repeating patterns. These patterns especially appear when different simulation goals are related to the same parameters. This is usually a valid assumption for every multiobjective optimization problem. Consequently, the list-based output of the Apriori algorithm is not suitable for efficiently analyzing the simulation dataflow as it can not represent these repeating patterns. These patterns would lead to additional effort in the simulation data farming process because the repeating relations would need to be analyzed multiple times.

We present a novel idea based on forest data structures in order to overcome these repeating patterns. The main idea is to generate for every simulation goal $\mathcal{G}$ a tree which denotes the level-1-itemset dataflow result of the ARM process for this particular simulation goal. Within these trees, repeating transactions will manifest as duplicated sub-trees which can be effectively pruned. Figure 3 additionally illustrates the first step of our knowledge discovery process, namely the forest generation.

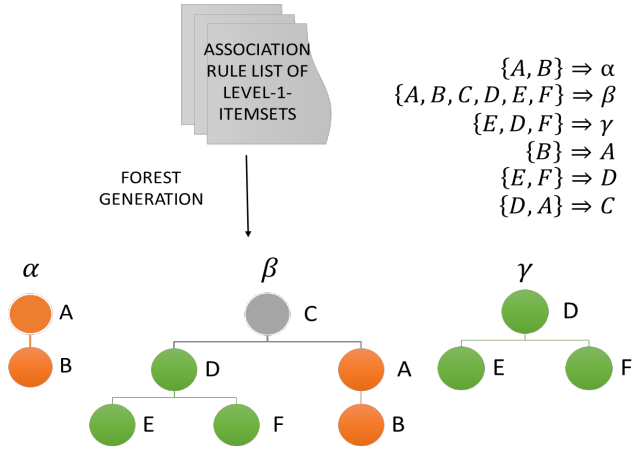

Figure 3: Level-1-itemsets are generated by the Apriori algorithm. The list based output is transformed into a forest structure which facilitates efficient simulation data farming. Repeating data access patterns from the simulation workflow result in prunable (green & orange) sub-trees of our forest.

Our utilization of the simulation transaction data as well as our forest data structure introduction reduces original computational complexity because we focus now in our simulation data farming phase only actual relationships:

$$\mathcal{O}((n^2 - n) \cdot g) \Rightarrow \mathcal{O}(k \cdot g) \tag{5}$$

where

k : number of simulation goal related input parameters with $k \leq n$

4.1 Cubic Spline based Simulation Data Farming & Sampling

In order to determine causal relations our knowledge discovery process needs to farm simulation data after the dataflow determination of the blackbox simulation. As stated before, a brute-force analysis would lead to computationally very expensive behavior of the complete knowledge discovery process.

In this section, we present the next step of our knowledge discovery process which involves our efficient farming of simulation data for the given parameter and simulation goal relations. As mentioned before, repeating transaction patterns of the simulation workflow will result in duplicated sub-trees in the forest.

As a consequence, we can simply prune a node n that does not have a causal relation to its parent node p. This relation is obviously valid for all trees in the forest. Consequently, all sub-trees of n can be removed of all forest trees. In the following, we present two algorithms to discover these causal relations as well as to minimize the sampling rate of the parameter space without losing objective value information. First, we define a spline approximation of the unknown objective function and second, we describe a recursive correlation analysis in Section 4.2.

Algorithm 3 ForestSampling

```
for 𝒯 ∈ ℱ do
  for 𝒩 ∈ 𝒯 do
    𝒮_goal = splineApprox(𝒩, 𝒯_goal)
    𝒮_parent = splineApprox(𝒩, 𝒩_parent)
    𝒞_goal = correlationAnalysis(𝒮_goal)
    𝒞_parent = correlationAnalysis(𝒮_parent)
    if 𝒞_parent < ϵ or 𝒞_goal < ϵ then
      Remove all subgraphs of 𝒩 in ℱ
    end if
  end for
end for
```

We assume that every relationship between a parameter $x = x_0, ..., x_k$ with parameter space k, unknown objective function f and utility value $y = y_0, ..., y_k$ can be formally represented as a continuous function $f_j\{x_i, ..., x_n\} \mapsto \mathcal{G}_j$. It would be possible to perfectly determine the behavior of f with respect to x by brute-force sampling the whole parameter space k. However, in real world applications k can be arbitrary large, therefore a brute-force sampling of the parameter space is infeasible. In order to overcome this challenge, we propose an approach based on cubic splines. A spline is a function that is piecewise defined by low-degree polynomials. Splines are often preferred in interpolation problems over higher-degree polynomial interpolation approaches because spline interpolation avoids the problem of Runge's phenomenon, i.e. oscillations that occur in interpolations between points when using high degree polynomials. Furthermore, even splines based on even polynomials can accurately approximate a given non-linear function.

The general idea of cubic splines is to represent the function by a different cubic function on each interval between data points. For n data points, the spline $S(x)$ is the function

$$S(x) = \begin{cases} C_1(x), x_0 \leq x \leq x_1 \\ C_i(x), x_{i-1} \leq x \leq x_i \\ C_n(x), x_{n-1} \leq x \leq x_n \end{cases} \quad (6)$$

where each C_i is a cubic function. The most general cubic function has the form

$$C_i(x) = a_i + b_i x + c_i x^2 + d_i x^3 \quad (7)$$

The main idea of our simulation data sampling and farming is to minimize the amount of samples n which are used to approximate the original behavior of f. In order to realize that, we iteratively approximate the unknown objective function f with a cubic spline. This spline is iteratively updated with more sampled data until the spline approximates f within a specified error degree. Algorithm 4 and Figure 4 illustrate this concept.

Algorithm 4 SplineApproximation

```
𝒟 = x_0, x_{k/2}, x_k
𝒮 = a spline based on 𝒟
ℛ = amount of remaining samples: k − 3
ℰ = list of rejections
while ℛ > 0 and ℰ < ϵ_rejections do
  𝒳 = evenly distributed x ∈ X
  𝒟 += 𝒳
  𝒴_sim = simulation result of 𝒳
  𝒴_spline = 𝒮(𝒳)
  if | 𝒴_sim - 𝒴_spline | < ϵ_deviation then
    ℰ += 𝒳
  end if
  𝒮 = rebuild spline based on 𝒟
  ℛ = ℛ - 1
end while
```

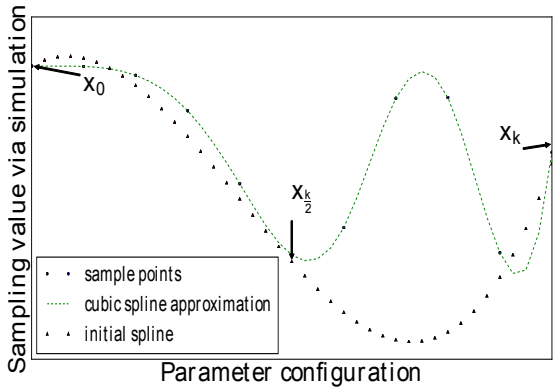

Figure 4: Spline approximation of a unknown objective function f: The spline is initialized with three initial sampled data points. The spline is iteratively updated until it approximates the unknown objective function within a certain error degree.

4.2 Recursive Correlation Analysis

When we have defined the cubic spline which represents the objective function f output $y = \{y_o, ..., y_n\}$ for a given parameter space $x = \{x_o, ..., x_n\}$, the next step is to determine whether or not a causal relation is present between x and y. This correlation analysis is needed as a spline approximation itself does not contain any correlation or causality information and can therefore approximate any given signal. Due to the fundamental property of self-containment of a simulation, confounding variables can be neglect as they would also be part of the workflow which would be uncovered by our ARM approach (see Section 4). Therefore, correlation alone can be used to determine the causal relation between x and y. In order to do so, we approximate the cubic spline representation s of f with segments which prove correlation. If the complete cubic spline s can be represented with such segments, we assume correlation and therefore causality between x and y.

Correlations between variables can be measured with the use of different indices (coefficients), such as the Pearson product correlation coefficient r [19]. It measures the linear correlation between two variables X and Y, giving a value

between +1 and -1, where +1 describes total positive correlation, 0 no correlation and -1 total negative correlation.

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2} \cdot \sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \quad (8)$$

where

n : number of elements in X resp. Y
x,y : elements of X and Y
$\bar{x}, \bar{y}$: sample mean $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ (analogously for $\bar{y}$)

However, non-linear objective functions (e.g. polynomials with degree ≥ 2) can not be correctly modelled with r. We therefore propose to recursively compute the Pearson coefficient. If the signal can not be described with r, we recursively split the signal in the middle and analyze the remaining signals. The causal relation is proven when the complete signal can be described with $\{r_k\}$, where k is the number of coefficients. Algorithm 5 and Figure 5 illustrates this concept:

Algorithm 5 RecursiveCorrelationAnalysis($\mathcal{S}$ spline)

$\mathcal{C}$ empty correlation segments
r_{xy} = pearson coefficient of $\mathcal{S}$[0,n]
if $|r_{xy}| >= r_{threshold}$ **then**
  $\mathcal{C}$ += $\mathcal{S}$[0,n]
else
  $\mathcal{C}$ += recursiveCorrelationAnalysis($\mathcal{S}[0, \frac{n}{2}]$)
  $\mathcal{C}$ += recursiveCorrelationAnalysis($\mathcal{S}[\frac{n}{2}$,n])
end if
return $\mathcal{C}$

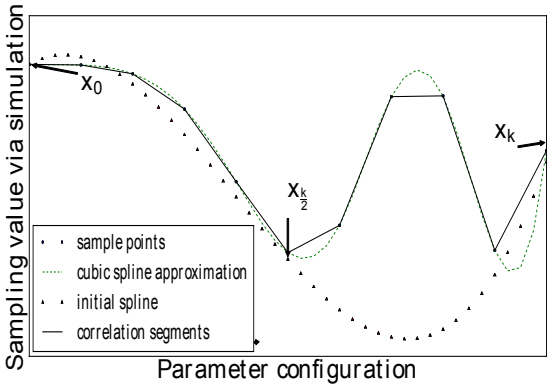

Figure 5: Result of the correlation analysis: Segments which prove linear correlation computed for the Spline representation of the objective function.

4.3 Determination of Pareto Information

After the determination of all causal relations between parameters $\{x_0, ..., x_p\}$ and simulation goals $\{g_0, ..., g_k\}$ (resp. objective functions $\{f_0, ..., f_k\}$) we can approximate the feasible design space for every parameter x_i. This feasible design space approximation can be used to efficiently compute suitable Pareto information for optimization algorithms.

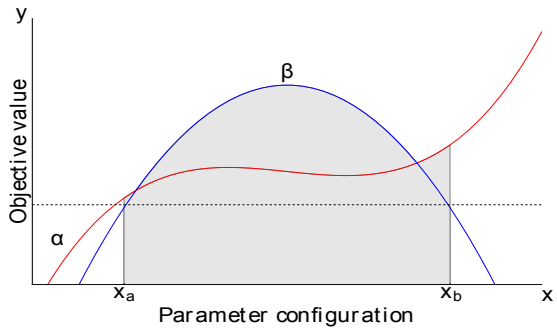

Figure 6: Approximation of feasible design space for a obtained parameter and simulation goal (α, β) relationship and given minimum objective value (dotted line): $x_a, ..., x_b$ determines the feasible parameter configurations.

The feasible design space constitutes a sub-set $\{\{x_i, ..., x_j\}, ..., \{x_n, ..., x_m\}\} \subseteq \{x_o, ..., x_k\}$ with $0 \leq i, i < j, n < m, j < n, m \leq k$. Figure 6 illustrates this concept for a parameter which is related to two simulation goals α, β.

Depending on the extracted design space, different outcomes are possible: singleobjective as well as multiobjective optimization problems. Singleobjective optimization problems can be directly solved by regular optimization algorithms. Multiobjective structures require Pareto information to solve conflicting objective functions. In the last case, we can further utilize our feasible design space approximation to generate Pareto gradient information.

An additional challenge appears if the same parameter contributes to different objective values at different time steps in the simulation (e.g. a fuel state/configuration in a car simulation which changes over time). Because of configurations like this, we prefer to define a spline of the objective function for each simulation time step individually. This results in a list of splines: $\{S(\{x_k\})_{t_0}, ..., S(\{x_k\})_{t_n}\} = \{\{y_k\}_{t_0}, ..., \{y_k\}_{t_n}\}$ for n time steps with:

$$S(\{x_k\})_{t_i} = \{y_k\}_{t_i} \quad (9)$$

where

t_i : simulation time
$\{x_k\}$: parameter space
$\{y_k\}$: objective values of the corresponding objective function

In the easiest case, every input parameter $x_i \in x_k$ would result in every time step in the same objective value y_i, hence $y_{i_{t_0}} = y_{i_{t_1}} = ... = y_{i_{t_n}}$. In this case, only one spline is needed to describe the relationship. In the other case, we define the deviations of x_i over the simulation time as

$$\alpha_t(x_i) = \{y_0, ..., y_m\} \quad (10)$$

where

$\{y_0, ..., y_m\}$: objective values for all simulation steps m

Additionally, we introduce a weighting function for this case in order to achieve smooth gradient transitions for the simulation behavior over the simulation time. We define this weighting as a Gaussian alignment of $\{y_o, ..., y_m\}$.

$$\omega_{x_i t} = \frac{e^{-k^2}\alpha_t(x_i) + ... + e^{-g^2}\alpha_{t_m}(x_i)}{m} \quad (11)$$

where

$k > m, k = 1, m$: number of simulation time steps
g : the Gaussian weight

In order to generate the required Pareto information, we transform this parameter space and objective space of the corresponding relations into a new representation which we call Pareto space.

The transformation into the Pareto space is implemented with $\omega_{x_i t}$:

$$\omega_{pareto}(x_i) = \frac{\sum \Theta_i(|\frac{o}{n} - \frac{o}{\sum_{j<n}^{j=0} f_j(x_i)} \cdot \omega_{x_i t}(x_i)|)}{k} \quad (12)$$

where

$1 \leq o \leq 100$: weighting factor
k : the number of non-empty utility functions
n : the number of related simulation goals
Θ : the Pareto weighting factor

We construct the Pareto space ω_{pareto} for all given Pareto goal weightings $\Theta_0, ..., \Theta_n$. Figures 7 and 8 additionally illustrate the definitions:

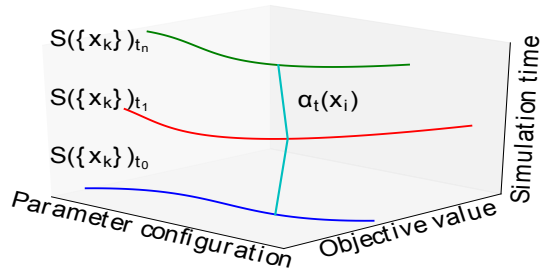

Figure 7: Graphs and deviation of a causal relationship between a parameter and a simulation goal.

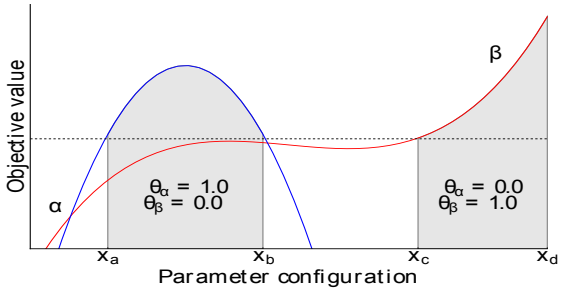

Figure 8: The Pareto gradient is determined by the amount of related simulation goals as well as Pareto weight Θ. The gradient information is computed for every sampling data point and constitutes our Pareto space. For every weighting Θ, a different Pareto space ($\{x_a, ...x_b\}$ and $\{x_c, ...x_d\}$) is constructed.

5. USE CASE STUDIES

We present two diverse use case studies from the fields of engineering and biology to illustrate the broad range of applications of our presented approach. First, we use Kepler's orbital mechanics which are used in space flight to describe celestial body and spacecraft movement. Second, we use the non-linear differential Lotka-Volterra equations [3] which are frequently used to describe the dynamics of biological systems in which to species interact, one as a predator and the other as prey. In both use case scenarios, the model behavior is unknown to our knowledge discovery process. Known to our approach are only the available input parameters as well as simulation goal measurements.

5.1 Spaceflight Orbit Optimization

5.1.1 Scenario

Usually spaceflight navigation solutions, especially autonomous interplanetary cruise flight, use optical measurements of reference bodies (e.g. Sun, Earth, Mars, Jupiter) to compute their position [2]. On-board optical systems take pictures of these reference bodies with respect to stars with known celestial locations. These images are used to compute the angular position of a spacecraft with respect to the reference bodies. These measurements are essential as spacecraft self-localization is needed throughout the complete mission [2]. Consequently, spacecraft trajectory calculation to target destinations has to consider possible reference bodies as well as optical system placing on the spacecraft for a given mission scenario.

The main idea of this use case is to compute a orbit for a interplanetary cruise flight to a target body in such a way that optical measurements to two reference bodies are guaranteed in order to ensure that spacecraft self-localization is possible.

5.1.2 Methodology

In celestial mechanics, Kepler's orbital elements can be used to uniquely identify a specific orbit in space. A Keplerian orbit is an idealized, mathematical approximation of an orbit for a particular time span. Each Kepler orbit is defined with six elements (see Figure 9), namely eccentricity e, inclination i, semimajor axis a, longitude of ascending node Ω, argument of periapsis ω and mean anomaly at epoch M.

Solving Keplers Equation $M(t) = E(t) - esin(E)$ for the eccentric anomaly $E(t)$ which defines the cartesian position of the specified orbit object can then be done with an appropriate method numerically, e.g. via Newton-Raphson iteration until $|E_n - E_{n-1}| \leq k$, where k defines the allowed deviation for the orbit (see Equation 13).

$$E_n = E_{n-1} - \frac{E_{n-1} - e \cdot sin(E_{n-1}) - M}{1 - e \cdot cos(E_{n-1}} \quad (13)$$

$\vec{r}$ is then defining the object position in cartesian coordinates with respect to the eccentric anomaly and support vectors $\vec{P}, \vec{Q}$:

$$\vec{r} = a \cdot (\vec{P} \cdot (cos(E) - e) + \sqrt{1 - e^2} \cdot \vec{Q} \cdot sin(E)) \quad (14)$$

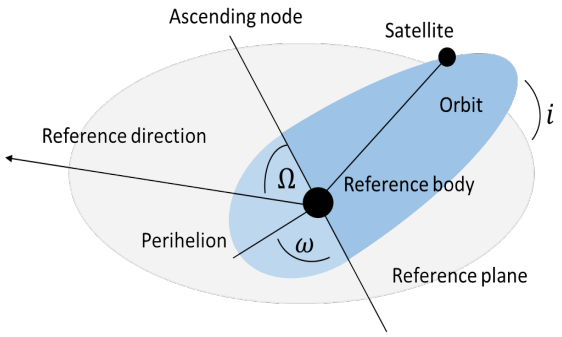

Figure 9: Illustration of the Keplerian orbital elements.

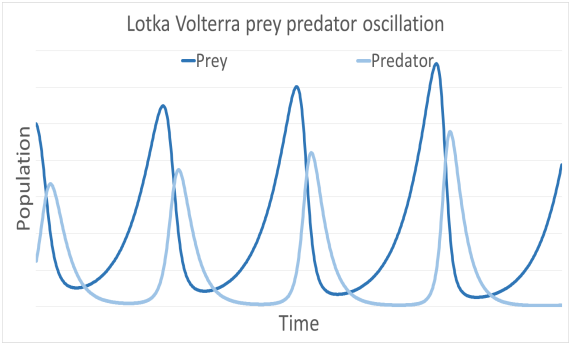

Figure 10: Illustration of the Lotka-Volterra equations: Periodic oscillation between preys and predators occur.

The two target bodies as well as the spacecraft are represented by their Kepler orbits around the Sun $k_1 = \{e, a, i, \omega, \Omega, M\}$, $k_2 = \{e, a, i, \omega, \Omega, M\}$, $k_{sc} = \{e, a, i, \omega, \Omega, M\}$. The required field of view of the sensor measurements are represented by

$$\cos\alpha = \frac{(\vec{r_{target}} - \vec{r_{sc}}) \cdot \vec{p}}{|\,(\vec{r_{target}} - \vec{r_{sc}})\,| \cdot |\,\vec{p}\,|} \leq t \tag{15}$$

where

$\vec{p}$: the aligned payload vector
t : the allowed angle of the payload field of view
$\vec{r_{sc}}$: the spacecraft position
$\vec{r_{target}}$: the target position

5.2 Lotka-Volterra Optimization

5.2.1 Scenario

The prey-predator system is a widely used simulation model of biological systems in which two species interact with each other. It consists of a dynamical non-linear system modeled by two differential equations, known as the Lotka-Volterra equations [3]. The equations model the evolution of two populations evolving in a common environment: preys and predators. Predators need to consume preys to survive, and preys spontaneously reproduce.

Due to the non-linear behavior of the Lotka-Volterra equations and further constraints (e.g. environmental conditions as the seasons which affect birth and death rate of the species) which can be added to the model, determining suitable input for observed real world ecosystem data is a challenging problem [22]. Therefore, the main idea of this use case is to determine a suitable input parameter set for the Lotka-Volterra equations in order to achieve a steady state between preys and predators for a given time span.

5.2.2 Methodology

The Lotka-Volterra model involves four parameters:

α : preys reproduction rate
β : preys death rate due to predators
δ : predators death rate in absence of preys
γ : predators reproduction rate according to consumed preys

The population evolution is given by these two differential equations:

$$\frac{dx(t)}{dt} = x(t)(\alpha - \beta y(t)) \tag{16}$$

$$\frac{dy(t)}{dt} = -y(t)(\delta - \gamma x(t)) \tag{17}$$

where $x(t)$ is the prey population at time t and $y(t)$ is the predator population at time t. Figure 10 further illustrates the behavior of the model.

6. EVALUATION

We implemented our knowledge discovery process and optimization approach in C++. We performed our experiments on a machine with Intel Core i7 quad-core processor with Hyperthreading enabled and 8GB of memory.

We applied different experiments to measure the performance as well as the quality of our approach. For the quality measurement, we used the use case scenarios described above. Both use case simulations were used to evaluate whether the computed Pareto gradient information are suitable for converging towards the Pareto front.

However, both scenarios are relatively small and can be hardly used to evaluate the performance of our approach. Hence, we additionally implemented a synthetic benchmark for performance measurements. We included three standard optimization algorithms: gradient descent, simulated annealing and evolutionary algorithms. The synthetic benchmark is based on blackbox simulations. We generated random objective functions for arbitrary simulation input parameters with mixed polynomials up to degree ten. These objective functions are further linked arbitrary times together with various simulation input parameters in order to generate multiobjective constraints.

Such multiobjective optimization problems do not have a single, accepted measure for solution quality [16], in contrast to single objective optimizations that have single global optimum, it is more complicated to measure the quality of any solution produced by a optimization algorithm.

We use the GD (Generational Distance) measurement [7] that provides an estimation of the distance of the current solution to the Pareto front. In other words, $GD = 0$ indicates that all solutions are placed on the Pareto front. We first compute the minimum Euclidean distance $\delta_i, i = 1, 2, ..., n_p$ of each solution where n_p is the number of solutions found.

The Generational Distance is then defined as:

$$GD = \frac{\sqrt{\sum_{i=1}^{n_p} {\delta_i}^2}}{n_p} \tag{18}$$

Figure 11 shows the mean average computation time for the implemented Association Rule Mining approach in our synthetic benchmark. Our approach generates level-1-itemsets for more than 10.000 simulation input parameters in less than a second (see Figure 11). Moreover, our approach is able to generate our forest data structure of up to 100 simulation goals for these 10,000 parameters in less than 100 milliseconds (see Figure 12). Consequently, our approach is able to analyze large scale simulations very effectively. Figure 13 shows the average sampling rate of our cubic spline based interpolation needed for successfully approximating an unknown objective function with parameter space $k = 100$. The sampling rate depends on the polynomial degree of the unknown objective function. We compared our spline based approach with an approach which randomly samples the parameter space. The results show that our approach needs less samples for determining the objective function.

All three optimization algorithms we tested directly benefit from our Pareto information. Here, we compared how close the algorithms optimize towards the Pareto front when using our provided Pareto gradient information or not. When using the provided Pareto gradient information from our approach, the algorithms find solutions closer to the Pareto front by up to 38% for gradient descent, 44% for simulated annealing and 81% for a evolutionary approach (see Figure 14).

Surprisingly, evolutionary algorithms benefit most from our Pareto information. We believe, that with an increasing number of conflicting goals, even our Pareto space inherits many local minima, which adversely affect gradient descent and simulated annealing.

We used our use case scenarios to measure the quality of our approach. In the prey predator system, our approach was successfully able to compute a suitable input parameter set in order to achieve a steady simulation state. Figure 15 shows initial and the optimized model behavior. In our space flight scenario, our approach obtained an optimized flight orbit of the spacecraft with respect to the needed sensor measurements (see Figure 16). In both use case studies, our approach computed suitable solutions based on our Pareto space and it achieved the desired simulation goal state.

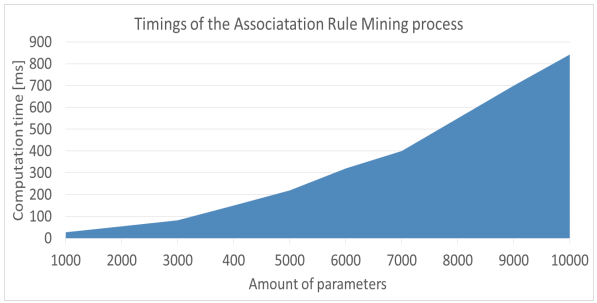

Figure 11: **Our customized Association Rule Mining approach is able to analyze several thousands of parameters for generating level-1-itemsets in less than a second.**

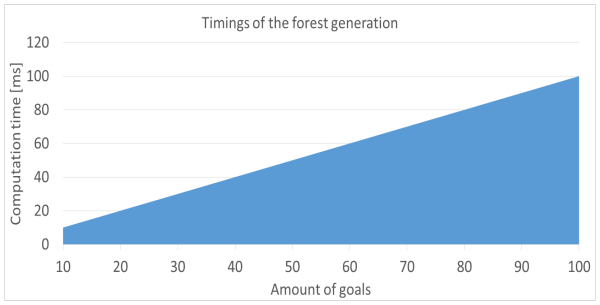

Figure 12: **Our forest generation algorithm is able to generate the corresponding tree structures for more than 100 goals in less than 100 milliseconds.**

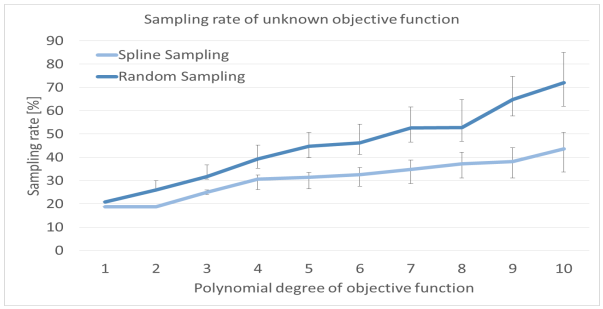

Figure 13: **Our cubic spline algorithm is able to efficiently approximate unknown objective functions with less than half of the available parameter space even for high degree polynomials.**

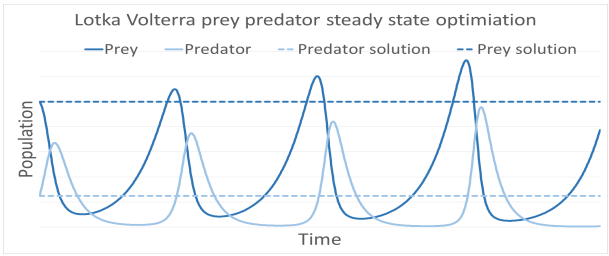

Figure 15: **Evaluation of the Lotka-Volterra use case study: Our approach is able to compute Pareto information in order to achieve a steady state in the population.**

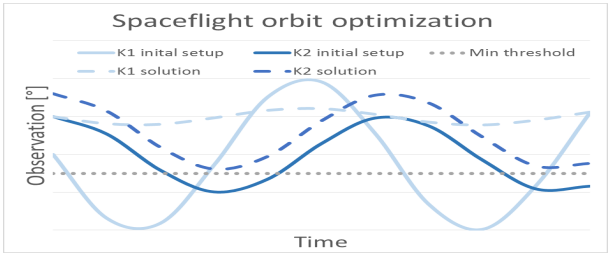

Figure 16: **Evaluation of the spacecraft flight use case study: Our approach is able to compute Pareto information in order to achieve the desired orbit for suitable observation possibilities.**

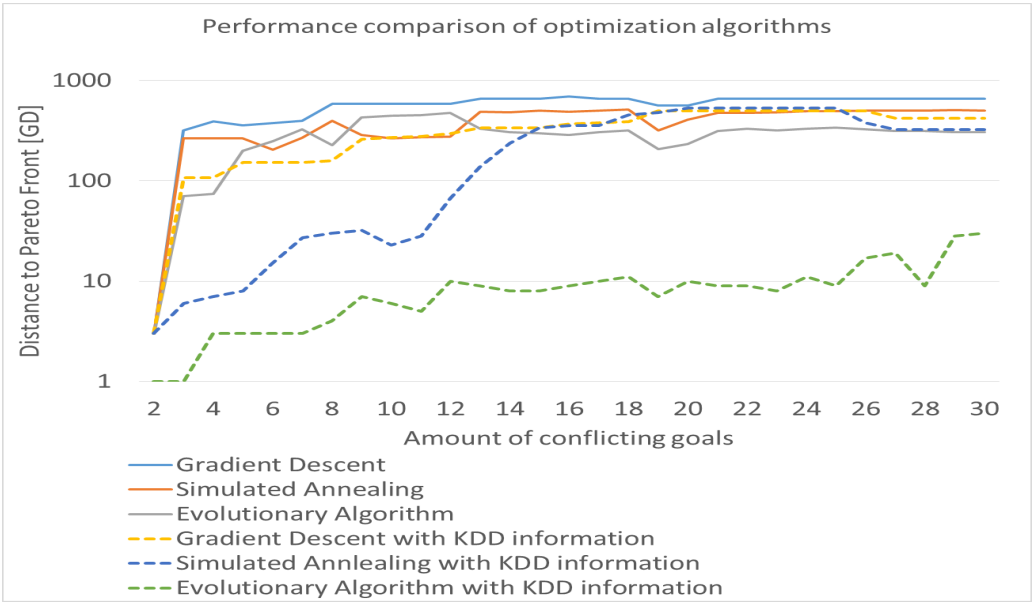

Figure 14: Our Pareto information directly benefit gradient descent, simulated annealing and evolutionary approaches and deliver results closer to the Pareto front.

7. CONCLUSION

We presented a novel knowledge discovery process for active model building of multiobjective optimization within deterministic blackbox simulations.

Our process automatically builds an active model between simulation input and simulation goals which is capable of

- approximating the feasible design space for a Pareto based optimization by uncovering unknown causal relations in large parameter sets between simulation input and model behavior which are assumed to be unknown non-linear objective functions.
- computing Pareto gradient information from this design space approximation which can be used in state-of-the-art multiobjective optimization solvers.

Our knowledge discovery process is completely autonomous and does not need any supervision from a simulation expert.

The results from our case studies and synthetic benchmarks show that our approach is able to analyze large scale simulations with tens of thousands of parameters in less than a second while providing suitable Pareto information for state-of-the-art multiobjective optimization algorithms. Due to its generality, our approach is applicable to a wide variety of simulation domains such as engineering design problems, including layout, design, and process optimization.

In the future, we would like to extend our approach to support intelligent, heuristic-based simulation data farming and sampling. In detail, our approach could sample directly in the direction of the Pareto front in order to improve the simulation data farming process. In order to realize this, a sampling heuristic is needed which should be in a direct feedback-loop with our proposed Pareto gradient optimization. Furthermore, an evaluation of our approach with standard optimization via simulation problems as provided by [29] is planned. Finally, it would also be beneficial if our approach could support stochastic simulations.

This would incorporate that our spline-based sampling technique needs to adapt to randomness in the sampling of objective values.

8. ACKNOWLEDGMENTS

This research is based upon the project KaNaRiA, supported by German Aerospace Center (DLR) with funds of the German Federal Ministry of Economics and Technology (BMWi) under grant 50NA1318.

9. REFERENCES

[1] Ajith Abraham, Lakhmi C. Jain, Robert Goldberg. Evolutionary Multiobjective Optimization: Theoretical Advances and Applications. *Springer Verlag*, 2005.

[2] Alena Probst, Graciela Peytavi, David Nakath, Anne Schattel, Carsten Rachuy, Patrick Lange, et al. Kanaria: Identifying the Challenges for Cognitive Autonomous Navigation and Guidance for Missions to Small Planetary Bodies. *International Astronautical Congress (IAC)*, 2015.

[3] Alfred J. Lotka. Elements of Physical Biology. *Williams and Wilkins*, 1925.

[4] Andreas Lattner, Joerg Dallmeyer, Ingo Timm. Learning Dynamic Adaptation Strategies in Agent-Based Traffic Simulation Experiments. *Ninth German Conference on Multi-Agent System Technologies (MATES)*, pages 77–88, 2011.

[5] Benjamin Wilson, David Cappelleri, Timothy W. Simpson, Mary Frecker. Efficient Pareto frontier exploration using surrogate approximations. *Optimization and Engineering*, pages 31–50, 2001.

[6] Catarina Dudas, Amos H. C. Ng, Henrik Bostroem. Post-Analysis of Multi-Objective Optimization Solutions Using Decision Trees. *Intelligent Data Analysis*, 19:259–278, 2015.

[7] David A. van Veldhuizen, Gary B. Lamont. Evolutionary Computation and Convergence to a Pareto Front. *Late Breaking Papers at the Genetic Programming Conference*, 1998.

[8] Dongkyung Nam, Cheol Hoon Park. Multiobjective Simulated Annealing: A Comparative Study to Evolutionary Algorithms. *International Journal of Fuzzy Systems*, 2, 2000.

[9] Eckart Zitzler, Lothar Thiele. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation*, 3:257–271, 1999.

[10] Eckhart Zitzler. Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications. 1999.

[11] Gary E. Horne, Theodore E. Meyer. Data Farming: Discovering Surprise. *Winter Simulation Conference*, pages 1082–1087, 2005.

[12] J. Westra, F. Dignum, V. Dignum. Guiding User Adaptation in Serious Games. *Agents for Games and Simulations II*, pages 117–131, 2011.

[13] Jack P.C. Kleijnen, Susan M.Sanchez, Thomas M. Cioppa. A User's Guide to the Brave New World of Designing Simulation Experiments. *INFORMS Journal on Computing (Summer 2005)*, 17:263–289, 2005.

[14] Jean-Antoine Desideri. Multi-Gradient Descent Algorithm (MGDA) for Multiobjective Optimization. *Comptes Rendus Mathematique*, 350:313–318, 2012.

[15] Joerg Fliege, B. Fux. Svaiter. Steepest Descent Methods for Multicriteria Optimization. *Mathematical Methods of Operations Research*, 3:479–494, 2000.

[16] John D. Siirola, Steinar Hauan, Arthur W. Westerberg. Computing Pareto Front Using Distributed Agents. *Computers and Chemical Engineering*, 29:113–126, 2004.

[17] Jong-Hyun Ryu, Sujin Kim, Hong Wan. Pareto Front Approximation With Adaptive Weighted Sum Method in Multiobjective Simulation Optimization. *Winter Simulation Conference*, pages 623–633, 2009.

[18] Jorge Nocedal, Stephen J. Wright. Numerical Optimization. *Springer Verlag*, 1999.

[19] Karl Pearson. Mathematical Contributions to the Theory of Evolution. *Philosophical Transactions of the Royal Society*, 187:253–318, 1896.

[20] Kazuyuki Sugimura, Shigeru Obayashi, Shinkyu Jeong. Multi-Objective Design Exploration of a Centrifugal Impeller Accompanied With a Vaned Diffuser. *ASME/JSME Joint Fluids Engineering Conference*, 2007.

[21] L. Jeff Hong, Barry L. Nelson. A Brief Introduction to Optimization via Simulation. *Winter Simulation Conference*, 2009.

[22] C. B. L. Pons. A Multi-Agent System for Autonomous Control of Game Parameters. *IEEE International Conference on Systems, Man and Cybernetics (SMC)*, 2013.

[23] Martin Liebscher, Katharina Witowski, Tushar Goel. Decision Making in Multi-Objective Optimization for Industrial Applications - Data Mining and Visualization of Pareto Data. *8th World Congress on Structural and Multidisciplinary Optimization*, 2009.

[24] M.C. Burl, D. DeCoste, B.L. Enke, D. Mazzoni, W.J. Merline, L. Scharenbroich. Automated Knowledge Discovery from Simulators. *Sixth SIAM International International Conference on Data Mining*, pages 82–93, 2006.

[25] Michael C. Fu. Optimization via Simulation: A Review. *Annals of Operations Research*, 53:199–248, 1994.

[26] Michael Painter, Madhav Erraguntla, Gary Hogg, Brian Beachkofski. Using Simulation, Data Mining, And Knowledge Discovery Techniques For Optimized Aircraft Enginee Fleet Management. *Proceedings of the Winter Simulation Conference*, pages 1253–1260, 2006.

[27] Niclas Feldkamp, Soeren Bergmann, Steffen Strassburger. Knowledge Discovery in Manufacturing Simulations. *ACM SIGSIM PADS*, pages 3–12, 2015.

[28] Patrick Lange, Rene Weller, Gabriel Zachmann. Multi Agent System Optimization in Virtual Vehicle Testbeds. *EAI SIMUtools*, 2015.

[29] R. Pasupathy. SimOpt: A Library of Simulation Optimization Problems. *Winter Simulation Conference*, pages 4075–4085, 2011.

[30] Rakesh Agrawal, Tomasz Imielinski, Arun Swami. Mining Association Rules between Sets of Items in Large Databases. *ACM SIGMOD Conference*, 1993.

[31] Ravindra V. Tappeta, John E. Renaud. An Interactive Multiobjective Optimization Design Strategy for Decision Based Multidisciplinary Design. *40th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference and Exhibit*, pages 78–87, 1999.

[32] Sanghamitra Bandyopadhyay, Sriparna Saha, Ujjwal Maulik, Kalyanmoy Deb. A Simulated Annealing-Based Multiobjective Optimization Algorithm: AMOSA. *IEEE Transactions on Evolutionary Computation*, 12:269–283, 2008.

[33] Songqing Shan, Gary G. Wang. An Efficient Pareto Set Identification Approach for Multiobjective Pptimization on Black-box Functions. *Journal of Mechanical Design 127*, 5:866–874, 2004.

[34] Sunith Bandaru, Kevin Deb. Automated discovery of vital knowledge from Pareto-optimal solutions: First results from engineering design. *IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8, 2010.

[35] Susan M. Sanchez. Simulation Experiments: Better Data, Not Just Big Data. *Winter Simulation Conference*, pages 805–816, 2014.

[36] Usama Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth. From Data Mining to Knowledge Discovery in Databases. *AI Magazine (Fall 1996)*, 17:37–54, 1996.

[37] William J. Frawley, Gregory Piatetsky-Shapiro, Christopher J. Matheus. Knowledge Discovery in Databases: An Overview. *Springer Verlag*, 1992.