GDS: Gradient based Density Spline Surfaces for Multiobjective Optimization in Arbitrary Simulations

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

ABSTRACT

We present a novel approach for approximating objective functions in arbitrary deterministic and stochastic multi-objective blackbox simulations. Usually, simulated-based optimization approaches require pre-defined objective functions for optimization techniques in order to find a local or global minimum of the specified simulation objectives and multi-objective constraints. Due to the increasing complexity of state-of-the-art simulations, such objective functions are not always available, leading to so-called blackbox simulations.

In contrast to existing approaches, we approximate the objective functions and design space for deterministic and stochastic blackbox simulations, even for convex and concave Pareto fronts, thus enabling optimization for arbitrary simulations. Additionally, Pareto gradient information can be obtained from our design space approximation. Our approach gains its efficiency from a novel gradient-based sampling of the parameter space in combination with a density-based clustering of sampled objective function values, resulting in a B-spline surface approximation of the feasible design space.

We have applied our new method to several benchmarks and the results show that our approach is able to efficiently approximate arbitrary objective functions. Additionally, the computed multi-objective solutions in our evaluation studies are close to the Pareto front.

Keywords

Objective function approximation; Knowledge discovery in simulation; multi-objective optimization; Spline interpolation; Simulation based optimization; Data mining; B-Spline surface

1. INTRODUCTION

Traditional simulation-based optimization approaches [17, 12] usually require pre-defined objective functions which directly describe the influence of all simulation input parameters on the specified simulation objectives (denoted as model behavior). An optimization toolset (e.g. [6]) uses these objective functions (e.g. ordinary differential equations) in order to find a local or global minimum which satisfies given constraints. As a consequence of the increasing complexity of state-of-the-art simulations, such objective functions are not always available. Even more, there are many technical complex systems whose long-term behavior can not be described by a set of equations (e.g. long-term behavior of autonomous systems in changing environments). This kind of simulation-based optimization problem is called blackbox simulation problem because the objective functions are unknown to both: the simulation engineer and optimization toolset.

Therefore, the model behavior analysis in these blackbox simulations and the determination of the valid design space, is either done manually by simulation experts or by an automated process. The goal in both approaches is to approximate the behavior of the objective functions in order to interpolate and extrapolate the model behavior. The manual analysis method is widely used [8], although it yields many disadvantages because it is based on subjective judgements from the simulation engineer [21]: the simulation expert usually takes an educated guess, based on his experience, which parameters might be influential on the simulation scope and varies these cleverly in multiple simulation runs in order to do both: approximate the feasible design and reduce the process complexity [19]. The resulting simulation results are later manually analyzed with respect to the given simulation constraints. This workflow can be partly supported (e.g. visualization of parameter sets, clustering analysis of results) by different tools [6, 19, 16] that can be introduced by the simulation expert. However, state-of-the-art blackbox simulation problems are further dominated by a multi-objective optimization problem (MOP) in which multiple and conflicting criteria have to be satisfied, which usually can not be determined by manual analysis [9]. Therefore, [8] refers to this as the error-prone "trial-and-error approach".

Consequently, recent approaches avoid a manual analysis and automatically analyze such multi-objective blackbox simulations via a knowledge discovery process which can compute suitable input configurations for a given simulation model without the need of an expert guiding this process. By definition, they incorporate some kind of data mining process which samples the simulation input parameter space with respect to known [19][16] or unknown [21] objective functions.

SIGSIM-PADS'17, May 24-26, 2017, Singapore.

ISBN 978-1-4503-4489-0/17/05…$15.00

DOI: http://dx.doi.org/10.1145/3064911.3064917

Unlike traditional approaches for solving MOP in blackbox simulations, these knowledge discovery processes in simulations are not limited to a static, pre-determined input dataset for model behavior and optimization. Instead, the simulation is used as a generator for new data by a simulation sampling process. This enables a knowledge discovery process to investigate the simulated model behavior in more detail and larger bandwidth [21] via its data mining scheme. This data mining scheme directly determines the efficiency and quality of the resulting objective function approximation because it defines the simulation sampling process.

However, state-of-the-art approaches do not support stochastic simulation behavior and are therefore restricted to deterministic simulations. Nevertheless, sophisticated simulations involve real-world scenarios which incorporate stochastic processes or properties. Approximating objective functions in stochastic simulations is more difficult and complex because the underlying noise of the stochastic process involves variations in the simulation and consequently in the data farming process. State-of-the-art studies model stochastic processes as deterministic ones which leads not only to inferior approximation of the objective functions but also to inferior solutions of the multi-objective optimization toolset as we will show in our evaluation. In this paper, we present a novel data mining approach which is directly based on above observations and the idea of knowledge discovery processes in simulations.

In detail, our approach is able to

- approximate objective functions (resp. the feasible design space) in arbitrary deterministic and stochastic blackbox simulations as B-spline surfaces,
- compute a Pareto gradient from the feasible design space approximation for concave, convex or interrupted Pareto fronts, which can be used with different optimization strategies,
- compute a Pareto solution from the feasible design space approximation via a hierarchical multi-agent system approach.

Another main advantage of our approach is that our B-spline surface based feasible design space approximation evaluation is computationally very fast and replaces costly simulation evaluations which are usually required. Consequently, our approach also delivers a performance boost when computing a solution for the given multi-objective optimization problem. Furthermore, our approach is very generic. It can be easily incorporated into existing simulation-based optimization approaches which use a knowledge discovery process (e.g. [21]). Even more, the computed Pareto solutions are close to the Pareto front for deterministic and stochastic simulations. Additionally, of our approach provides optimization strategies. These strategies can be used by state-of-the-art multi-objective optimization solvers in order to investigate a larger bandwidth of the simulated model behavior.

Our presented approach is immensely important for simulation applications because simulation technology in general bases its core competence on the usage of simulation results. Our work is designed to extract as much valuable information as possible from a simulation system and its results, thus, increasing the usefulness of sophisticated simulations.

2. RELATED WORK

Approximating objective functions in simulation-based optimization scenarios has attracted increasing interest in the last decade. Mostly, such approximations are implemented as a sophisticated data mining approach within a complex knowledge discovery (KD) process.

The research can be classified into two groups: KD approaches aiming at single or multi-objective optimization problems. All presented KD approaches have in common, that they only support deterministic simulations. Current KD approaches for single-objective optimization problems reduce the optimization problem, e.g. to a single function f which updates the simulation state x with parameter set θ via $x_{k+1} = f(x_k, \theta)$ [16]. This approach reduces the complexity of the optimization process by finding a suitable set of points θ in which a pre-defined simulation state is achieved. Likewise, other approaches, such as [18], neglected multi-objective simulation properties. The approach applied a linear regression to all input parameters $x_i, ...x_n$ for a specific simulation objective state y. This results in a linear model which was used to describe the input configuration space. This model was further used in a clustering approach in order to find the most influencing parameters. The above studies can not be applied to multi-objective based simulations in which the simulation model is governed by a set of (possible) contradicting functions because they do not concern such structures within the simulation. Similar approaches, are either restricted to a small number of simulation input parameters due their quadratic runtime [2] or are highly depending on the simulation expert supervising the KD process [19].

In summary, all above mentioned studies focused on building passive models between simulation input and objective-related simulation output while minimizing the simulation parameter scope or by focusing on single-objective linear simulation models in deterministic simulations. These passive models describe the simulation without enabling interpolation or extrapolation of the simulated model behavior for simulation-based optimization purposes. They deliver coarse granularity parameter relationship information which can not be used to approximate the feasible design space nor to compute a Pareto gradient information (e.g. gradient information of the analyzed data with respect to the Pareto front), especially for stochastic simulations.

In contrast to the above studies, [21] introduced a KD process which builds an active model between simulation input and simulation output. It is able to uncover hidden relationships between simulation input and unknown objective functions. It approximates the feasible design space which is suitable for solving multi-objective optimization problems. Other KD approaches based on multi-objective simulations [25, 11, 15, 5] focussed on extracting additional information from pre-determined concave Pareto sets or analzying these sets within the simulation. Consequently, they can not be used to approximate the feasible design space nor to compute a Pareto solution itself. Concluding, all above studies are restricted to deterministic simulations which leads to several disadvantages as stated before.

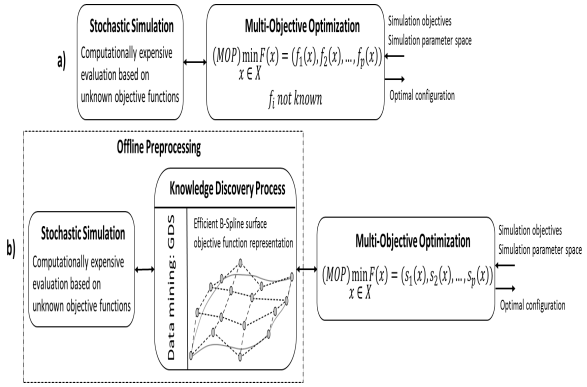

Figure 1: Our approach introduces an efficient objective function approximation within a knowledge discovery process via novel B-spline surface representations.

3. OUR APPROACH

In this work, we present our gradient based density spline surface (GDS) algorithm (see Figure 1). GDS is able to approximate arbitrary unknown objective functions in deterministic and stochastic blackbox simulations, for convex and concave as well as interrupted Pareto fronts. GDS consists of three main concepts:

- B-spline surface representation of the relationship space (see Section 3.1),
- density based clustering of objective function samples which determines the noise behavior of the stochastic simulation (see Section 3.2),
- gradient based sampling of the parameter space which reduces the required amount of samples (see Section 3.3).

The result of GDS is an efficient approximation of the objective functions as a set of B-spline surfaces which can be used in various simulation based optimization scenarios, such as complex multi-objective optimization problems. This efficient approximation is computationally very inexpensive compared to usually very costly simulation evaluations. It can be further effectively utilized in optimizaton toolsets. We present the application in our multi-agent system optimization approach as a use-case study (see Section 3.4).

3.1 Relationship Definition

We assume that every relationship between a parameter $C = c_0, ..., c_k$ with parameter space k and objective value $O = o_0, ..., o_k$ with objective space k can be formally represented as a continuous function $f : C, T \mapsto O = f(c, t) \mapsto o_{\mathcal{O}}$ which maps the parameter space to a given simulation objective $\mathcal{O}$ with its objective function space at a given time step $t \in T$ of the simulation. It would be possible to perfectly determine the behavior of f with respect to C by brute-force sampling the whole parameter space k with $s \geq k$ samples. However, in real world applications k can be arbitrary large or continuous and the simulation evaluation computationally very expensive. Therefore, a brute-force sampling of the parameter space is infeasible. Consequently, it is necessary to reduce the amount of needed samples: $s \ll k$.

Overall, the relationship constitutes a large three- dimensional cartesian space $\mathbb{R}^3$ (spanned by C, O and T), which we denote as relationship space (see Figure 2).

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

$$S(x) = \begin{cases} F_1(x), x_0 \leq x \leq x_1 \\ F_i(x), x_{i-1} \leq x \leq x_i \\ F_n(x), x_{n-1} \leq x \leq x_n \end{cases} \quad (1)$$

where each F_i is a cubic function.

The most general cubic function has the form

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

In sophisticated simulations, the same parameter configuration will contribute 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: $S_{t_o}(C), ..., S_{t_n}(C) = O_{t_0}, ..., O_{t_n}$ for n simulation time steps with:

$$S_{t_i}(C) = O_{t_i} \quad (3)$$

where

t_i : simulation time
C : parameter space
O : objective values of the corresponding objective function

We use these splines to formulate a cubic B-spline surface:

$$C(u) = \sum_{i=0}^{n} p_i N_{i,3}(u), 0 \leq u \leq 1 \quad (4)$$

$$s(u, v) = \sum_{i=0}^{m} \sum_{j=0}^{n} P_{ij} N_{i,3}(u) N_{j,3}(v), 0 \leq u, v \leq 1 \quad (5)$$

where $P_{ij} (i = 0, 1, ..., m; j = 0, 1, ..., n)$ are the control points of the surface which are determined by $S_{t_o}(C), ..., S_{t_n}(C) = O_{t_0}, ..., O_{t_n}$ via a uniform coverage of the splines. u, v are the knot vectors in the direction of u or v and $N_{i,3}(u), N_{j,3}(v)$ is the B-spline basis (see Figure 2).

This B-spline approximation replaces the unknown objective function $f : C, T \mapsto O = f(c, t) \mapsto o_{\mathcal{O}} = s(c, t) \mapsto o_{\mathcal{O}}$ and is efficiently used to define the required gradient information (see Section 3.4) for optimization purposes.

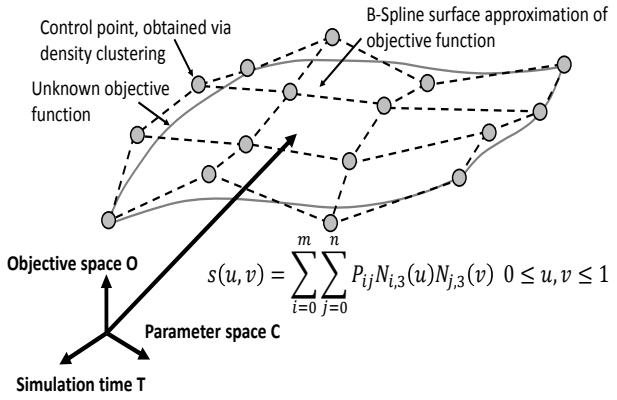

Figure 2: **B-Spline surface representation of the three-dimensional space constructed by simulation input parameter C, simulation time T and objective function space O.**

3.2 Density Splines

In order to enable a precise B-spline surface approximation of the relationship space, the splines which define $s(u, v)$ must be very close to the unknown objective function. However, as stated above, stochastic simulations are governed by diverse noise behavior which makes it hard to approximate these unknown objective function. Therefore, we define every spline sampling point (c, o) via a density clustering from n simulation samples.This density clustering detects noise outliers and therefore enables our splines to approximate the unknown objective function more precisely.

In order to incorporate this density clustering for the unknown noise distribution of the objective function, we extend the general spline definition (see Equation 1). Every spline is defined with a triple, consisting of the cubic function F_i, variance of the computed density cluster θ and certainty of measurement p.

In detail, every F_i is constructed with the centre point of the most dense cluster of every simulation sample set per sampling configuration of the simulation. In order to incorporate the simulation noise behavior, every centre point is associated with the variance θ of the corresponding cluster. Our spline definition will interpolate some of the objective values due to the gradient-based sampling of the parameter space (see Section 3.3) because we sample only a small subset of the parameter space. This means that, by definition, some approximated objective values are more likely precise (based on simulation samples) than others (based on the spline interpolation). Therefore, p indicates whether or not the resulting approximated objective value is interpolated resp. close (in parameter space) to a drawn simulation sample.

Therefore, for n sampling points, the spline $S_{t_i}(c)$ is the function:

$$S_{t_i}(c) = \begin{cases} (F_1(c), \Theta(\psi), p(c,\theta)), c_0 \le c \le c_1 \\ (F_i(c), \Theta(\psi), p(c,\theta)), c_{i-1} \le c \le c_i \\ (F_n(c), \Theta(\psi), p(c,\theta)), c_{n-1} \le c \le c_n \end{cases} \quad (6)$$

where

$\Theta(\psi)$: variance $\frac{1}{k-1} \cdot \sum_{i=0}^{k} (o_i - \bar{o})^2$

$p(c, \theta)$: certainty of measurement

$$\begin{cases} 1, \text{ if } c \in \theta \\ 1 - \frac{|c - c_j|}{k}, \text{ otherwise} \end{cases}$$

Ω : Dbscan core cluster of ψ

ψ : n samples of c $\{(c, o_i), ..., (c, o_n)\}$ with c $\in \theta$

θ : sampling configurations

$F_i(x)$: cubic function based on θ: $a_i + b_i x + c_i x^2 + d_i x^3$

c_j : closest previous sample point $\in \theta$ to c

The sampled objective values are clustered with the Dbscan (density-based spatial clustering of applications with noise) data clustering algorithm [14] (see Figure 3). Dbscan has several advantages with respect to other clustering approaches (such as [7, 3, 13]) because

- it does not require a specification on the expected cluster amount,
- it can find arbitarily shaped clusters,
- it has a notion of noise which makes it robust to outliers.

This density clustering is important because it enables a more accurate approximation of the unknown objective function via its detection of noise outliers. Therefore, just averaging ψ is insufficient. Even more, the computed Dbscan clusters can be used to retrieve the standard deviation and variance of the measurement. We further utilize this information in our optimization process in order to investigate arbitrary multi-objective problems from different optimization perspectives (see Section 3.4).

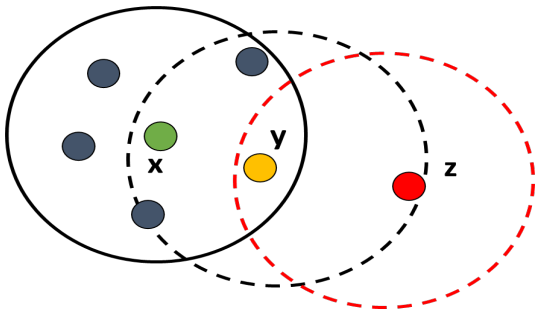

Figure 3: **Illustration of the Dbscan algorithm: A data point is inside a dense region (core point, x), on the edge of the region (boundary point, y) or in a spare region (noise point, z). The example is given for at least six neighbours. Adapted from [14].**

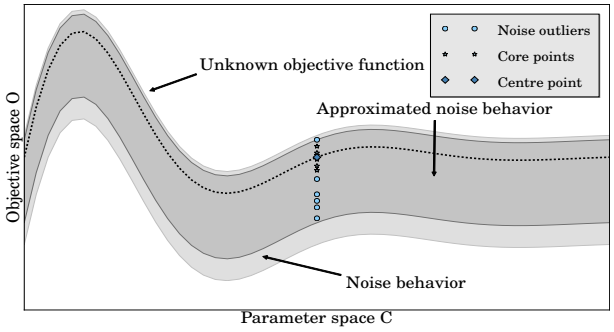

Figure 4: Approximating an unknown objective function with our density spline: Noise outliers are detected and the cluster centre point is used to construct the spline. Subsequently, the variance Θ of the clustering is propagated through the spline.

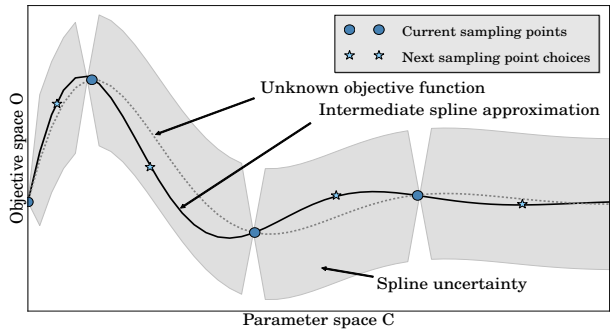

Figure 5: Spline approximation of an unknown objective function f: The spline is iteratively updated until it approximates the unknown objective function within a certain error degree. The spline uncertainty p is propagated through the spline.

3.3 Gradient-based Sampling

The main idea of our gradient-based simulation data sampling is to minimize the amount of samples s (see Section 3.1) which are required approximate the original behavior of f. In order to do so, we iteratively approximate the unknown objective function f with our spline definition for a certain simulation time t. This spline is iteratively updated with more sampled data until the spline approximates f within a specified error degree.

In order to minimize the amount of required samples, we utilize a property of spline-based interpolation. When interpolating with splines, the spline can change drastically when updated with new sample points at interpolated gradient minima and maxima. This is due to the fact that splines ensure that the first and second derivative of the spline will match at the knot points. Therefore, it is desirable to determine the spline gradient while approximating the unknown objective function in order to find large spline gradient changes. These gradient changes are used to draw a new sample from the parameter space which will more likely change the adjacent spline knots and therefore reduce the amount of required samples due to the aforementioned inherent behavior of spline interpolation (see Algorithms 1,2 and Figure 5). After successfully approximating the unknown objective function, we combine the variances of all clusters for each spline in order gain an accurate approximation for the noise distribution of the relationships. Given two clusters, C_1 and C_2, with their respective mean and variance of their largest cluster, $\bar{X_1}, \bar{X_2}$ and ${S_1}^2, {S_1}^2$ with n_1, n_2 observations, we compute the combined variance (see Equation 7). We iteratively apply this formula on all clusters of each spline and retrieve the overall variance of the spline approximation.

Algorithm 1 Objective function approximation via density splines splineApprox(amount of samples $\mathscr{n}$, parameter $\mathscr{C}$ with space $\mathscr{k}$, spline error threshold ϵ, simulation time $\mathscr{t}$)

$\mathscr{D} = c_0, c_{\frac{k}{2}}, c_k \in \mathscr{C}$, sampling configurations
$\mathscr{F}$ = simulation results of $\mathscr{D}$ with $\mathscr{n}$ samples
$\mathscr{K}$ = Dbscan clusters of $\mathscr{F}$
$\mathscr{S}$ = spline based on $\mathscr{D}, \mathscr{K}$
$\mathscr{R}$ = amount of remaining samples: $\mathscr{k} - 3$
$\mathscr{E}$ = empty list of rejections
while $\mathscr{R} > 0$ **and** $\mathscr{E} < \epsilon_{rejections}$ **do**
  $\mathscr{d}$ = gradientConfiguration($\mathscr{S}$,$\mathscr{D}$)
  $\mathscr{D}$ += $\mathscr{d}$
  $\mathscr{O}$ = empty list of simulation results
  for $\mathscr{n}$ samples **do**
    $\mathscr{O}$ += simulation result of $\mathscr{d}$ at $\mathscr{t}$
  end for
  $\mathscr{o}_{spline} = \mathscr{S}(\mathscr{d})$
  $\mathscr{T}$ = largest Dbscan cluster of $\mathscr{O}$
  $\mathscr{o}_{sim}$ = centre of $\mathscr{T}$
  $\mathscr{o}_{\Theta}$ = variance of $\mathscr{T}$
  $\mathscr{o}_p = 1$
  if $\mid \mathscr{o}_{sim}$ - $\mathscr{o}_{spline} \mid < \epsilon_{deviation}$ **then**
    $\mathscr{E}$ += $\mathscr{d}$
  end if
  $\mathscr{S}$ = rebuild spline based on $\mathscr{D}, \mathscr{O}$
  $\mathscr{R} = \mathscr{R}$ - 1
end while
return $\mathscr{S}$

Algorithm 2 Sampling of the parameter space based on gradient information: gradientConfiguration(current spline definition $\mathscr{S}$, sampled configuration $\mathscr{D}$)

$\mathscr{c}$ = return configuration
$\mathscr{t}$ = maximum threshold: 0
for $\mathscr{d}_1, \mathscr{d}_2, \in \mathscr{D}$ **do**
  $\nabla_{\mathscr{d}_1} = \dot{\mathscr{K}}(\mathscr{d}_1)$
  $\nabla_{\mathscr{d}_2} = \dot{\mathscr{K}}(\mathscr{d}_2)$
  if $|\nabla_{\mathscr{d}_1} - \nabla_{\mathscr{d}_2|} > \mathscr{t}$ **then**
    $\mathscr{c} = \frac{\mathscr{d}_1 + \mathscr{d}_2}{2}$
    $\mathscr{t} = |\nabla_{\mathscr{d}_1} - \nabla_{\mathscr{d}_2}|$
  end if
end for
return $\mathscr{c}$

$$S_c{}^2 = \frac{n_1 {S_1}^2 + n_2 {S_2}^2 + n_1(\bar{X_1} - \bar{X_c})^2 + n_2(\bar{X_2} - \bar{X_c})^2}{n_1 + n_2} = \frac{n_1[{S_1}^2 + (\bar{X_1} - \bar{X_c})^2] + n_2[{S_2}^2 + (\bar{X_2} - \bar{X_c})^2]}{n_1 + n_2} \quad (7)$$

where

$$X_c{}^2 = \frac{n_1 \bar{X_1} + n_2 \bar{X_2}}{n_1 + n_2} \quad (8)$$

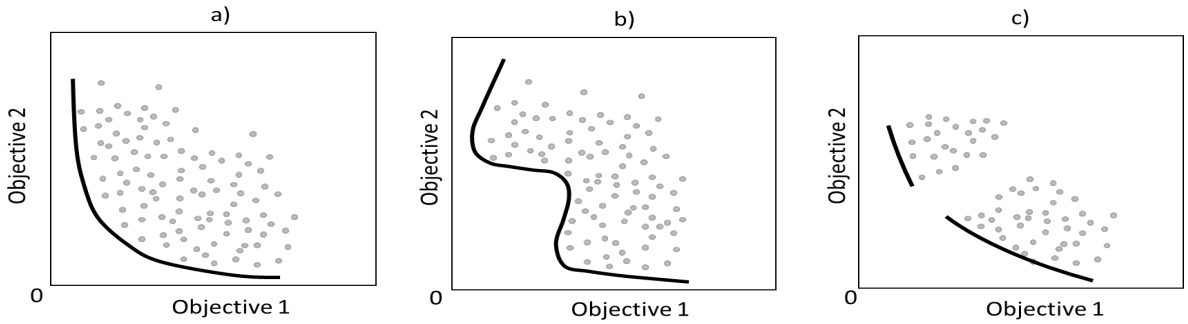

Figure 6: **Pareto optimal solutions of multi-objective dominate every other possible solution: Concave (a), convex (b) or interrupted Pareto fronts (c) can occur. Our approach efficiently approximates all possible frontiers.**

3.4 Multi-Objective Optimization

Today, simulation models are dominated by a multi- objective optimization problem (MOP) because many real world problems involve decisions based on multiple and conflicting criteria [9, 21]. The goal of multi-objective optimization is to determine best trade-off solutions (so-called Pareto solutions) among these criteria. These multi-objective optimization problems can be found in many situations, for example, in product design where several criteria must be simultaneously satisfied [4, 23, 24]. We define MOP according to [9, 21]: 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:

$$(MOP) \max_{x \in X} F(x) = (f_1(x), f_2(x), ..., f_p(x)) \tag{9}$$

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 $f_j(x), j = 1, ..., p$. In these scenarios, the goal of MOP is to identify a subset of the Pareto optimal points ($\mathscr{P}*$) which is able to represent the Pareto front or to compute a single trade-off solution $x \in \mathscr{P}*$ (see Figure 6) [21]. 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. $F(x_u)$ dominates $F(x_v)$ (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 [10], 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.

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 [9].

Within simulation-based multi-objective optimization problems, engineers are interested in several different optimal solutions, e.g. which are reliable in many scenarios or which maximize the objective function for certain aspects. Therefore, our approach enables a Pareto solution of a multi-objective optimization problem with three different optimization strategies which determine different feasible design spaces while maintaining Pareto efficiency:

- Compliance strategy: Determination of the parameter space which maximizes or minimizes the objective function.
- Reliability strategy: Determination of the parameter space which maximizes the sampling probability.
- Closeness strategy: Determination of the parameter space which minimizes the clustering variance.

Due to our relationship definition, we can substitute the unknown objective functions $f_j(x), j = 1, ..., p$ from Equation 9 with our B-spline surface approximation:

$$\begin{aligned} &(MOP) \max_{x \in X} F(x) = (f_1(x), f_2(x), ..., f_p(x)) \\ &\qquad\qquad \Leftrightarrow \\ &(MOP) \max_{c,t \in C,T} F(c,t) = (s_1(c,t), s_2(c,t), ..., s_p(c,t)) \end{aligned} \tag{10}$$

Our strategies determines a different feasible design space within above definition, namely a sub-set $\{\{c_i, ..., c_j\}, ..., \{c_n, ..., c_m\}\} = C_{Strategy} \subseteq \{c_o, ..., c_k\} = C_k$ with $0 \leq i, i < j, n < m, j < n, m \leq k$. In detail, the compliance strategy (see Equation 11) maximizes objective function above a given minimum threshold, m, for all $t \in T$. In contrast to this, the reliability strategy (see Equation 12) and closeness strategy (see Equation 13) either maximize the probability p or minimize the variance Θ of each measurement.

$$C_{compliance} = \{c_i | \forall s_q(c_i, T).F \geq m\} \quad (11)$$

$$(c_i, \overline{p_i}) = (c_i, \frac{\sum_{q=0}^{q=k} \sum_{n=0}^{t} s_q(c_i, t_n).p_i}{k}) \quad (12)$$
$$C_{reliability} = max_p\{(o_0, p_0), ..., (o_k, p_k)\}$$

$$(c_i, \overline{\Theta_i}) = (c_i, \frac{\sum_{q=0}^{q=k} \sum_{n=0}^{t} s_q(c_i, t_n).\Theta_i}{k}) \quad (13)$$
$$C_{closeness} = min_\Theta\{(o_0, \Theta_0), ..., (o_k, \Theta_k\}$$

Depending on the strategy, different sub-sets are determined. Each sub-set is transformed into a feasible design space ω, depending on the multi-objective constraints of the specific configuration parameter, namely the set of objective functions which are influenced by the paramter:

$$\omega_i(c,t) = \sum_{p=0}^{p=k} \Theta_p \cdot |\frac{o}{n} - s_p(c,t) \cdot \frac{o}{\sum_{q=0}^{q=k} |t_q - s_q(c,t)|}| \quad (14)$$

where

$0 \leq o \leq 1$	: weighting factor
k	: the number of related objective functions of i
t	: the objective threshold
Θ	: the Pareto weighting factor
c	: configuration from corresponding strategy

Consequently, our approach is able to find either a qualitative solution (see Equation 11), a reliable solution (see Equation 12) or the most dense solution (see Equation 13) that can be directly used in order to investigate the multi-objective problem from different perspectives.

The Pareto gradient $\nabla\omega$ is defined with unit vectors i, j, k which span the feasible design space:

$$\nabla\omega_{pareto} = \frac{\partial\omega}{\partial c}i + \frac{\partial\omega}{\partial t}j + \frac{\partial\omega}{\partial o}k \quad (15)$$

Figure 7 illustrates this concept for the compliance strategy in a simple two-dimensional example.

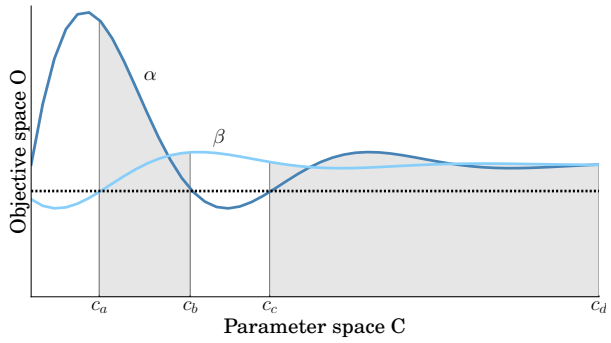

Figure 7: Approximation of the feasible design space for a parameter space, simulation objectives (α, β) and given minimum objective value (dotted line): $(c_a, c_b), (c_c, c_d)$ determines the feasible parameter configurations with $\Theta_\alpha = \Theta_\beta = 1$.

4. MAS BASED OPTIMIZATION

We present in this section how our feasible design space approximation and optimization strategies can be incorporated into a powerful optimization toolset for finding a Pareto solution. The optimization system proposed here is based on a hierarchical multi-agent system (MAS) which aims at dynamically tuning all given input configuration parameters with respect to the approximated feasible design space. Such hierarchical MAS have already proven their feasibility for solving multi-objective optimization problems such as [6, 20]. Our main idea is that every agent introduces a part-wise modelling (single and multi-objective constraints per input parameter) of the problem and its behavior and communication to other agents is used to solve the global (multi-objective optimization) problem.

In the following, we describe at first the required MAS infrastructure with its modular agent organizations and their relationships to the feasible design space approximation. Following this, we explain the input and output data as well as communication structure of the agents. At last, we outline the adaptation solving process with its negotiation mechanisms and how multi-objective problems can be solved.

Our MAS is composed of several agent organizations. Each of these organizations aims at optimizing a subset of configuration parameters to one or more simulation objectives, each one represented by our feasible design space approximation.

These agent organizations are defined per specified simulation objective and consist of a hierarchy of two agent types: objective- and negotiation-agents. For each identified input parameter, one objective-agent is defined. Therefore, one objective-agent can belong to several agent organizations. The goal of every defined objective-agent is to maximize or minimize every attached simulation objective under Pareto constraints. Several optimization constraints arise because of the underlying multi-objective optimization problem. Therefore, a negotiation-agent is defined for every specified objective. The goal of every negotiation agent is to manage requests between the objective-agents in order to satisfy the existing multi-objective constraints between the objective-agents.

Figure 8 illustrates this agent organization concept with respect to the overall proposed approach. The input and output data as well as communication structure of these agents is described in the next section.

4.1 Parameters, Objectives and Utilities

Given a multi-objective optimization problem, as defined in Equation 10, we can uniquely identify every adjustable input configuration parameter C with its valid range. These configuration parameters and the corresponding B-spline surface based feasible design space approximation constitute the input of our multi agent system. Additionally, we define the objectives and utilities of our MAS based control system.

Each objective-agent strives for maximizing or minimizing each single-objective optimization problem of its attached input parameter under multi-objective constraints. Therefore, each objective-agents computes several objective values, one for each attached objective.

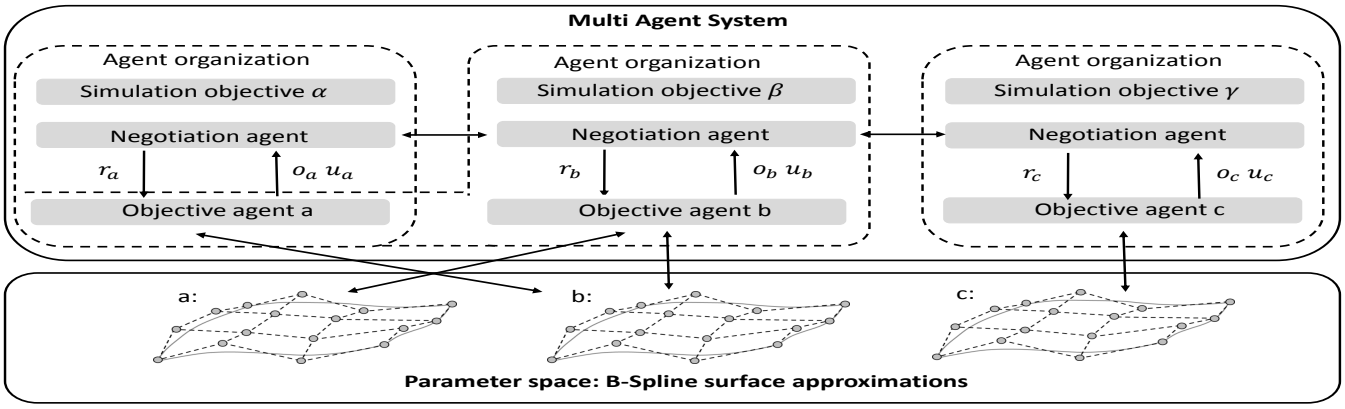

Figure 8: Our MAS based optimization approach for a mixed objective problem statement (one multi-objective objective (β) and two single-objective problems (α, γ) with three input parameters): Each agent organization optimizes the parameter set for one objective. Negotiation agents handle requests between the objective-agents in order to effectively find the optimal parameter configuration.

The set of all agent objectives OBJ is defined as follows:

$$OBJ = \{o_1 ... o_n\} \rightarrow \{0, 1\}$$
$$o_i = \omega(c, t) \tag{16}$$

where

ω : corresponding approximated feasible design space

In addition to the objectives to be satisfied, our MAS also considers the possible utility when changing the given parameter with respect to the current negotiation state among the agents. Consequently, we introduce a utility function. It is used to calculate the difference between current Pareto solution and highest achievable single-objective solution. This utility value is then later used by the negotiation-agent to select the most appropriate action.

The set of all utility values is defined as follows:

$$UTL = \{u_1 ... u_n\} \rightarrow \{0, 1\}$$
$$u_i = s_i(c, t) - \omega(c, t) \tag{17}$$

where

s_i : corresponding approximated objective function
ω_i : corresponding approximated feasible design space

Therefore, the aim of our MAS based optimization system is to minimize UTL while maximizing OBJ. In other words, it is to tune all the parameters so all the constraints and objectives are satisfied.

In order to implement this efficiently in an agent-based organization, we define requests which are shared between the agents. These requests are used to lower or higher the Pareto weight or objective value threshold from Equation 14 if any agent has partially solved one objective:

$$REQ = \{(\Theta_o, t_o), ...(\Theta_n, t_n)\}$$
$$\Theta_i \rightarrow \{0, 1\} \tag{18}$$
$$t_i \rightarrow \{0, 1\}$$

where

Θ_i : corresponding Pareto weight for the associated objective
m_i : corresponding objective value threshold for the associated objective

4.2 Solving Process Principle

When designing a multi-agent system based optimization process, the focus is set on agent behaviors and communications in order to cover the isolated parts of the global problem which each agent models. Each of our objective-agent tackles an isolated sub-problem (finding a solution to its attached single or multi-objective constraints) and emergence is used to solve the overall (multi-objective optimization) problem. Therefore, the solving process is distributed among all objective-agents via negotiation-agents. Consequently, the definition of objective- and negotiation-agent behaviours is also one of the key aspects of our multi-agent system and is described hereafter.

- Negotiation-agents monitor the objectives and utilities of all corresponding objective-agents of their agent organization and distribute requests between objective-agents: In the first step, they update (see below) the attached objective-agents if they have open requests. In this step, each objective-agent may achieve a new Pareto solution. In the second step, they collect new requests from each objective-agent and group them according to the multi-objective Pareto satisfaction:
  - a) if the objective is completely satisfied, it requests a change for the Pareto weight $\Theta = 0$ for every other attached objective-agent.
  - b) if the objective is partially satisfied, it requests a change for the objective threshold t in order of magnitude of current objective satisfaction for every other attached objective-agent.

  At last, the negotiation-agent will forward the requests (change in objective threshold t or Pareto weight Θ) of those objective-agents with the highest utility value.
- An objective-agent has a rather simple behaviour: it computes the objective and utility value for every attached objective for the current Pareto configuration (objective thresholds, Pareto weights and parameter configuration) based on our feasible design space approximation. It updates these values every time a request for change in Pareto weighting or objective threshold is received from an negotiation-agent.

5. EVALUATION

We implemented our GDS algorithm 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 and quality of our approach within several synthetic benchmark scenarios. These synthetic benchmarks are based on generated blackbox simulations. Consequently, the objective functions of the simulation are unknown to all evaluated algorithms. The synthetic benchmarks compared the performance of our GDS algorithm to the approximation approach from [21] and different clustering (Dbscan, k-means) and sampling approaches (uniform, random, gradient). In order to perform this evaluation, we generated two different types of random objective functions based on polynomials (f_p) and Gaussian functions (f_g). Furthermore, we added noise terms (N) for ten different noise distributions in order to obtain a stochastic simulation behavior. a, b, c, p, q are the known scalar values for each corresponding function:

$$f_p(c,t) = \sum_{i=0}^{n} a_i (c-p)^i + \sum_{j=0}^{m} b_j (t-q)^j + N \qquad (19)$$

$$f_g(c,t) = \sum_{i=0}^{n} a_i e^{-\frac{(c-b_i)^2}{2c_i^2}} + \sum_{j=0}^{m} b_j e^{\frac{(t-b_j)^2}{2c_j^2}} + N \qquad (20)$$

Based on above equations, we have evaluated our approach with more than 100 different versions of f_p and f_g in order to obtain a profound evaluation. These synthetic benchmarks have been supplemented by a qualitative evaluation in order to evaluate whether or not our B-spline surfaces can be used for optimization purposes.

We compared the mean approximation error for approximating an unknown polynomial objective function with a 30% noise variance (see Figure 9) and a relationship space of 10,000. Our GDS approach is able to outperform all its competitors up to a factor of four. This performance boost also increases with the noise variance of the unknown object function (see Figure 10). This evaluation shows that the non-clustering approaches perform worse than any clustering-based approach. Even more, the error variance of our GDS approach is the smallest, especially when comparing to the uniform sampling approach without clustering [21] as well as for the random sampling. Therefore, it can be observed that a clustering is inevitable in order to approximate the unknown objective function accurately.

In addition to the mean error evaluation, Figure 10 shows the mean average approximation error for an increasing noise behavior of the unknown objective function. It can be clearly observed that the clustering approaches adapt very well to an increased noise behavior of the unknown objective function while all other approaches clearly decrease in their performance.

Furthermore, all non-clustering based approaches need more samples than the clustering based competitors, when comparing their performance for the same required approximation error threshold (see Figure 11).

In this context, Figure 12 shows the impact of the sampling amount with respect to the clustering analysis. It can be observed that only a few samples (≤ 12) per simulation configuration are required in order to precisely ($\leq 10\%$) approximate the given unknown polynomial objective function. Overall, these evaluations strongly emphasize the quality improvement of our GDS approach with respect to its competitors.

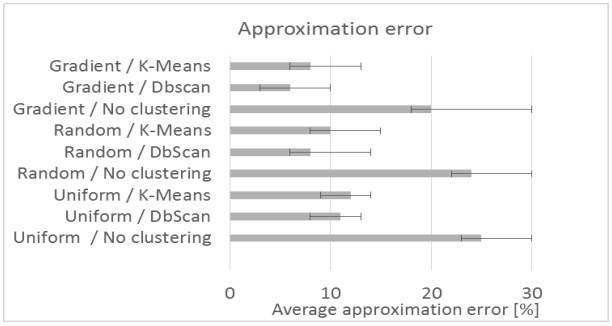

Figure 9: Our approach (gradient/Dbscan) outperforms its competitors: It approximates unknown objective functions with less error and smaller error variance.

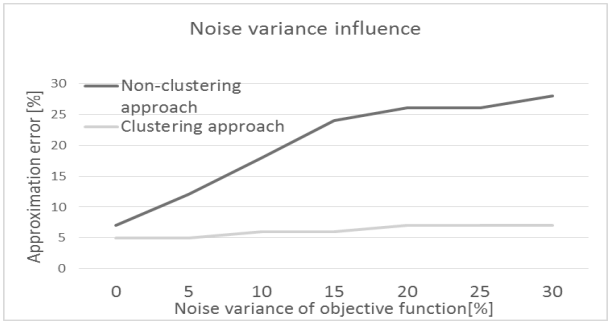

Figure 10: Clustering based approaches do not suffer as much as non-clustering approach from the objective function noise behavior.

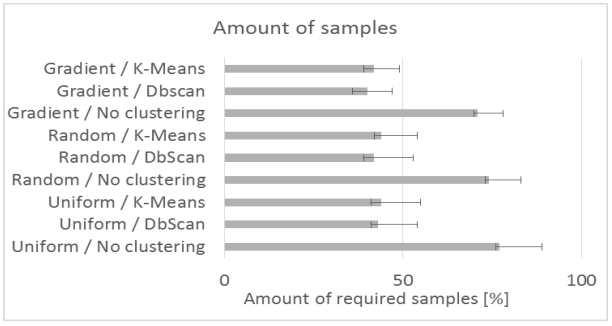

Figure 11: Our approach (gradient/Dbscan) requires less samples than its non-clustering based competitors. It further delivers both, best sampling variance and best approximation error.

Furthermore, Figure 13 depicts the relationship between polynomial degree of the objective function and the GDS mean error of the objective function approximation. Surprisingly, our GDS approach is almost not affected by the polynomial degree of the objective function. Therefore, even complex objective functions can be precisely approximated by our GDS approach.

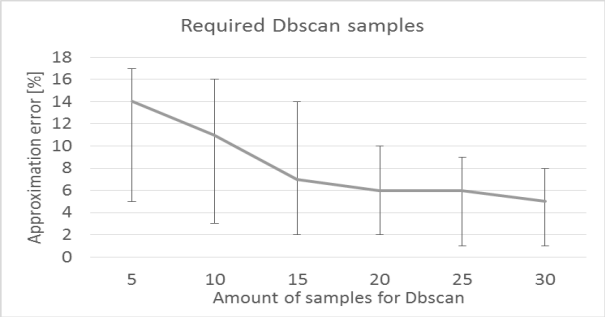

Figure 12: Only a few samples ($\leq$ 12) are required in order to efficiently ($\leq$ 10 % error) approximate the noise distribution of the stochastic simulation.

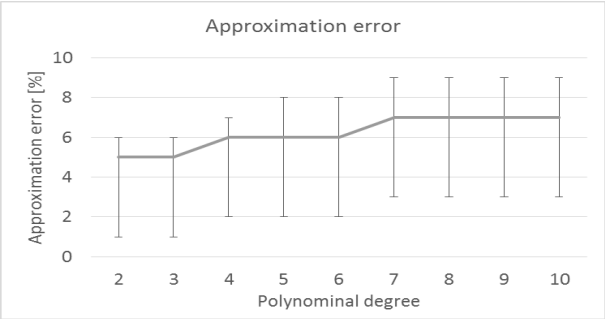

Figure 13: Effect of the polynomial degree of the unknown objective function on the mean error of our approach: The degree has almost no influence as the mean error varies around 6%.

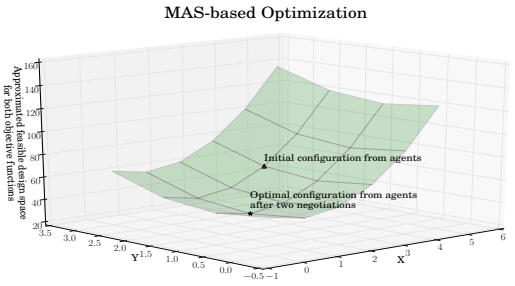

Figure 14: Evaluation of our use case study: Our agents are directly initialized at the single-objective solution and converge fast to the multi-objective solution.

For deterministic simulations, all presented approaches perform very similar and obtain approximation errors less than 1% for arbitrary polynomial objective functions.

Finally, Figure 14 depicts the results from our proposed MAS-based optimization. For this evaluation, we considered the following standardized test function for multi-objective optimization problems from Binh and Korn [26] as a use case study. We further added noise terms to the functions in order to obtain a stochastic behavior:

$$
\begin{aligned}
f_1(x,y) &= 4x^2 + 4y^2 + N \\
f_2(x,y) &= (x-5)^2 + (y-5)^2 + N \\
&s.t. \\
g_1(x,y) &= (x-5)^2 + y^2 \leq 25 \\
g_2(x,y) &= (x-8)^2 + (y+3)^2 \geq 7.7
\end{aligned}
\tag{21}
$$

In this use case study, we approximate f_1, f_2 with our GDS algorithm and deliver the B-splines as input to our MAS. Two advantages from our approach can be observed in this use case study (see Figure 14): First, the initial guess from the agents is directly the single-objective solution of the problem, indicating that the approximated feasible design space is close to the Pareto front. This enables a much faster optimization process because our MAS requires less negotiations for converging to the correct solution. This reduces the probability of converging to a local instead of a global minimum. Second, already after a few negotiations (in this use case study: two negotiations) the objective-agents reached the Pareto front and returned one optimal configuration.

In summary, Table 1 shows a detailed overview of our synthetic benchmarks for both test functions f_p and f_g. It can be seen that our GDS algorithm outperforms its competitors for all given noise distributions in mean error.

6. CONCLUSION

We presented our novel GDS approach for approximating unknown objective functions in arbitrary deterministic and stochastic blackbox simulations which are governed by a multi-objective optimization problem.

Our approach is capable of

- approximating objective functions (resp. the feasible design space) in arbitrary deterministic and stochastic blackbox simulations,
- computing Pareto gradient from the feasible design space approximation for concave, convex or interrupted Pareto fronts, which can be used with different optimization strategies,
- computing a Pareto solution from the feasible design space approximation via a hierarchical multi-agent system approach.

Even more, our approach can be easily integrated into existing knowledge discovery processes.

The results from our benchmarks show that our approach is able to analyze stochastic simulations with large parameter spaces while precisely approximating arbitrary unknown objective functions. Furthermore, the resulting optimization solutions are close to the Pareto front. 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 further evaluate our approach with standard optimization via simulation problems using the SimOpt library [22]. Additionally, we would like to extend our approach with other interpolation approaches: we could analyze the unknown objective function with several approximations (linear, polynomial, spline) in parallel. This would lead to specific approximation types per simulation objective which could further minimize the approximation error. Another interesting idea would be to replace our B-spline surface concept with a high dimensional input

space. Here, we would need to extend our B-spline surface concept to B-spline volumes. These B-spline volumes could be directly used for high-dimensional optimization. At last, we would like to incorporate GPGPU programming into our data mining approach. We believe that such massively parallel implementation can be efficiently used to analyze large-scale simulations.

7. 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.

8. REFERENCES

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

[2] 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.

[3] A.P. Dempster, N.M. Laird, D.B. Rubin. Maximum Likelihood from Incomplete Data via the EM algorithm. *Journal of the Royal Statistical Society*, 39:1–38, 1977.

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

[5] 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.

[6] Dario Izzo. PYGMO and PYKEP: Open Source Tools for Massively Parallel Optimization in Astrodynamics. *International Conference on Astrodynamics Tools and Techniques (ICATT)*, 2012.

[7] David Arthur, Sergei Vassilvitskii. k-means++: The Advantages of Careful Seeding. *SODA '07 at 8th ACM-SIAM symposium on Discrete algorithms*, 1027-1035, 2007.

[8] 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.

[9] 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.

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

[11] 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.

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

[13] Linli Xu, James Neufeld, Bryce Larson, Dale Schuurmans. Maximum Margin Clustering. *Advances in Neural Information Processing Systems*, 17:1537–1544, 2005.

[14] Martin Ester, Hans-Peter Kriegel, Joerg Sander, Xiaowei Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. *AAAI Press*, pages 226–231, 1996.

[15] 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.

[16] M.C. Burl, D. DeCoste, B.L. Enke, D. Mazzoni, W.J. Merline, L. Scharenbroich. Automated Knowledge Discovery from Simulators. *6th SIAM International International Conference on Data Mining*, pages 82–93, 2006.

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

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

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

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

[21] Patrick Lange, Rene Weller, Gabriel Zachmann. Knowledge Discovery for Pareto based Multiobjective Optimization in Simulation. *ACM SIGSIM PADS*, 2016.

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

[23] 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.

[24] 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.

[25] 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.

[26] Thanh Binh, Ulrich Korn. An Evolution Strategy for the Multiobjective Optimization. *2nd International Conference on Genetic Algorithms*, pages 23–28, 1996.

Sampling:	**Uniform**						**Random**						**Gradient**					
Clustering:	**Det.**		**Dbscan**		**K-Means**		**Det.**		**Dbscan**		**K-Means**		**Det.**		**Dbscan**		**K-Means**	
	$\bar{e_p}$	$\bar{e_g}$	$\bar{e_p}$	$\bar{e_g}$	$\bar{e_p}$	$\bar{e_g}$	$\bar{e_p}$	$\bar{e_g}$	$\bar{e_p}$	$\bar{e_g}$	$\bar{e_p}$	$\bar{e_g}$	$\bar{e_p}$	$\bar{e_g}$	$\bar{e_p}$	$\bar{e_g}$	$\bar{e_p}$	$\bar{e_g}$
Probability mass functions																		
Binominal, $t = 9.0, p = 0.5$ $P(i\|t,p) = \binom{t}{i} \cdot p^i \cdot (1-p)^{t-i}$	15.8	15.9	10.71	10.8	13.86	13.96	16.68	16.71	10.55	10.59	13.42	13.56	15.65	15.72	8.67	8.71	11.34	11.41
Geometric, $k = 0.3$ $p(i\|k) = k \cdot (1-k)^i$	15.66	15.71	10.61	10.67	13.34	13.58	16.54	16.68	10.33	10.4	13.23	13.32	15.52	15.76	8.62	8.69	11.56	11.63
Pascal, $k = 3.0, p = 0.5$ $p(i\|k,p) = \binom{k+i-1}{i} \cdot p^k \cdot (1-p)^i$	15.6	15.72	10.58	10.61	12.26	13.37	16.27	16.32	10.32	10.4	13.67	13.7	15.65	15.86	8.53	8.6	11.23	11.38
Uniform, $a = 0.1, b = 9.0$ $p(x\|a,b) = \frac{1}{b-a}$	15.33	15.56	10.29	10.41	13.44	13.49	16.02	16.1	10.07	10.12	13.72	13.79	15.3	15.53	8.32	8.45	11.67	11.75
Poisson, $\mu = 0.1$ $p(x\|\mu) = \frac{\mu^i}{i!} e^{-\mu}$	15.59	15.81	10.54	10.61	12.19	12.55	16.42	16.51	10.3	10.38	13.21	13.32	15.56	15.62	8.56	8.61	11.85	11.93
Probability density functions																		
Cauchy, $a = 5.0, b = 1.0$ $p(x\|a,b) = \frac{1}{\pi \cdot b \cdot [1+(\frac{x-a}{b})^2]}$	15.1	15.3	10.44	10.48	12.42	12.56	16.24	16.28	10.25	10.3	13.33	13.42	15.15	15.25	8.47	8.52	11.24	11.4
Chi-squared, $n = 3.0$ $p(x\|n) = \frac{1}{\Gamma(\frac{n}{2}) \cdot 2^{\frac{n}{2}}} \cdot x^{\frac{n}{2}-1} \cdot e^{-\frac{x}{2}}$	15.7	15.9	10.61	10.63	12.11	12.46	16.28	16.43	10.4	10.48	13.56	13.61	15.7	15.78	8.62	8.7	10.87	10.96
Fisher-F, $m = 2.0, n = 2.0$ $p(x\|m,n) = \frac{\Gamma(\frac{m+n}{2})}{\Gamma(\frac{m}{2}) \cdot \Gamma(\frac{n}{2})} \cdot \frac{\frac{mx}{n}^{\frac{m}{2}}}{x \cdot (1+\frac{mx}{n})^{\frac{m+n}{2}}}$	15.35	15.51	10.84	10.88	12.71	12.83	16.35	16.48	10.73	10.79	13.56	13.6	15.13	15.31	8.75	8.81	11.74	11.83
Normal, $\mu = 5.0, \sigma = 2.0$ $p(x\|\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{(x-\mu)^2}{2\sigma^2}}$	15.56	15.62	10.55	10.63	11.98	12.68	16.4	16.53	10.39	10.44	13.67	13.72	15.52	15.59	8.48	8.54	11.52	11.6
Exponential, $\lambda = 3.5$ $p(x\|\lambda) = \lambda e^{-\lambda x}$	15.65	15.7	10.62	10.67	13.46	13.66	16.36	16.48	10.34	10.41	13.73	13.78	15.6	15.72	8.63	8.78	11.64	11.67

Table 1: Synthetic performance comparison overview: Our GDS approach (grey) outperforms its competitors for all noise distributions (mean error $\bar{e}$ in %)