University of Zurich[UZH]

Bayesian Optimization for Best Response Computation in Combinatorial Auctions

Bachelor's Thesis
September 19, 2018

Marius Högger
of Zurich, ZH, Switzerland

Student-ID: 13-920-053
marius.hoegger@uzh.ch

Advisor: **Vitor Bosshard**

Prof. Dr. Sven Seuken
Department of Informatics
University of Zurich
http://www.ifi.uzh.ch/ca

Acknowledgements

I would like to thank my advisor, Vitor Bosshard, for guiding me through this Bachelor's Thesis, for numerous constructive meetings, for sharing knowledge and for giving great feedback. I would also like to thank Vitor for giving me Beta-access to the repository of the CA-BNE algorithm which I extended and which provided a lot of inspiration for my implementation.

My gratitude goes to Prof. Dr. Sven Seuken for making it possible to write this thesis in his Computation and Economics Research Group and also for making me attentive to the topics Computation and Economics.

Last, but not least I am grateful to Joel Barmettler, Janine Dössegger, David Lehnherr, Jacqueline Riedi and Yannik Zemp for proofreading and providing valuable feedback.

Abstract

The Bayes-Nash equilibrium (BNE) solution to combinatorial auctions (CAs) bears much information helping to understand allocation problems since it represents a stable situation where non of the bidders included would want to deviate. This thesis explores the possibility of using Bayesian optimization (BO) to reduce the number of expected utility evaluations when searching for the pointwise best response which is used for finding the BNE. This research is based on the algorithm presented in [Bosshard et al., 2017]. Comparing BO with pattern search (PS) on the univariate LLG-domain, the reduction of the costly evaluations is estimated and the influence of the variance of the Monte Carlo (MC) integration used for the calculation of the expected utility, is examined. Finally, the quality of the results when BO is used, is verified by comparing them against the known analytical results. Confirming that when using BO, equilibria of similar quality can be found with fewer evaluations, this thesis still highlights certain reservations concerning the effectiveness of BO for finding best responses in the LLG domain.

Zusammenfassung

Um Zuordnungsprobleme besser zu verstehen, kann das Bayes-Nash Equilibrium (BNE) von kombinatorischen Auktionen untersucht werden da dieses eine stabile Situation wiedergibt in welcher sich keiner der beteiligten Bietern umentscheiden würde. Diese Arbeit untersucht inwiefern sich die Bayes'sche Optimierung (BO) eignet um die Anzahl Auswertungen der erwarteten Nutzenfunktion zu reduzieren. Diese Auswertungen werden verwendet um das BNE zu finden. Die Forschung in dieser Arbeit basiert auf dem von [Bosshard et al., 2017] präsentierten Algorithmus. Es wird erforschen inwiefern BO die Anzahl Auswertungen reduzieren kann und in welcher Hinsicht die Varianz der Monte Carlo (MO) Integration, welche zur Bestimmung des erwarteten Nutzens verwendet wird, einen Einfluss auf die Bestimmung des Maximums der Nutzenfunktion hat. Dabei wird BO mit dem "pattern search" (PS) Algorithmus auf dem eindimensionalen LLG-Bereich verglichen. Abschliessend überprüfen wir die Qualität des mittels BO gefundenen Resultates, indem wir dieses mit der bekannten analytischen Lösung vergleichen. Diese Arbeit bestätigt, dass durch die Verwendung von BO die Anzahl Auswertungen reduziert werden kann, ohne dass, die Qualität der Lösung zu sehr zu beeinträchtigen wird. Dennoch erhebt diese Arbeit einige Zweifel gegenüber der Effektivität von BO im LLG-Bereich.

Table of Contents

1

Introduction

Combinatorial Auctions (CAs) are an active field of research since many of the real world's allocation problems can be reflected on them. Contrary to simpler auction settings, CAs do allow preference dependencies between different items which are considered to be more realistic models. In fact, a real bidder can have many different valuations for a good, depending on what other goods are available to it. Allowing a bidder to report values for bundles of items maps these context-aware valuations to an auction friendly representation. A simple example that demonstrates why realistic items have dependent values is described in section 2.1.
CAs are often used by governments to allocate transmission rights and frequencies for telecommunication [Cramton, 2013]. These two domains by themselves are multi-billion dollar domains but there are abounding applications for CAs, therefore it is not surprising that a lot of research effort is spent on this field. Simple and well-known mechanisms like the VCG mechanism[1], which is strategy-proof, behave unfortunate on the setting of CAs, can lead to inferior revenue and are vulnerable to coalitions [Ausubel et al., 2006]. More promising are mechanisms that are core-selecting. Being core-selecting prevents the exposure to a coalition and ensures that there is no losing bidder who would be willing to pay more than the payment the mechanism assigns to the winner [Day and Raghavan, 2007]. However, such mechanisms are no longer strategy-proof, meaning there are strategies that can be applied to gather greater utility where bidding the own value (being truthful) is no longer the best option. As a consequence studying the mechanism at truth, so where all bidder report their true value, is not meaningful because there is no guarantee that this situation is ever reached. Therefore, the mechanism has to be studied in equilibrium, since this is the stable situation in which after knowing the outcome, no bidder would want to change its reported value. For a repeated auction it can be assumed that eventually such an equilibrium is reached.

Usually only CAs are considered, where bidders do not know the other bidders' values, which is the so-called incomplete information settings. The sole information a bidder has about other bidders' values is the rough distribution of valuations. Having to calculate with distribution instead of explicit values makes the whole computations more complex but it is still possible to calculate analytical results, as shown by [Ausubel and Baranov,

[1] A mechanism developed by Vickrey, Clarke and Groves

2018]. Analytical results have only been derived for simple and low dimensional problems, which lead to the approach of using computational means to simulate the different bidders reacting to the previous actions of the other bidders and thus adjusting their strategies and finally finding the Bayes-Nash equilibrium (BNE). These simulations include calculating the best response for many different values the bidders can have for the items.

Calculating the best response is equivalent to solving an optimization problem since the goal is to find the one bid that optimizes the expected utility, given the expected valuations of the other bidders and given a specific value of the bidder itself. Practically, the calculation of the expected utility can be performed using the Monte Carlo (MC) integration. Meaning that for a given bid[2] of the bidder at focus, many different combinations of the other bidders' valuations are drawn according to their distribution. Having explicit valuations available, the outcome can be calculated simply by applying the auction's specific rules which determine the outcome, meaning which bidders win which items and how much they have to pay for them. Knowing the true value of the bidder and the outcomes, the utilities can be derived directly. In the end all the intermediary results are gathered and averaged in one final expected utility. The more combinations of valuations of the other bidders' are drawn the more accurate is the expected utility. Consequently the process of calculating the expected utility for a given bid already uses a substantial computational effort. But the goal is to find the bid that yields the highest expected utility. This whole process can be interpreted as an evaluation of a function that assigns an expected utility to a given bid. As stated above the best-response calculation is equivalent to optimizing this function. Since a single evaluation is very expensive, it is beneficial to be able to optimize the function with the least evaluations possible.

1.1 Goal

Bayesian optimization (BO) has been proven to use only a few evaluations to find the maximum of a function. Therefore, this thesis aims to research the possibility of applying BO for the optimization of the expected utility function in the best-response calculation. The research is done in combination with the algorithm[3] derived in [Bosshard et al., 2017] which from now on will be referenced under the name CA-BNE. [Bosshard et al., 2017] uses pattern search (PS) to optimize the expected utility function in the best-response calculation. Therefore, PS acts as a comparison benchmark for the BO. The ambition is to find out whether the BO approach is able to produce the same results as PS and whether it can produce them with less computational effort. Further, this thesis investigates how different aspects of the BO have an impact on the performance

[2] reported value which does not need to be the true value of the bidder

[3] The Beta-release of the algorithm from [Bosshard et al., 2017] was accessed via the private repository [Bosshard et al., 2018]. Bosshard confirmed verbally that the algorithm will be released publicly in the near future.

and produced results of the whole CA-BNE.

1.2 Research Methods

To achieve these goals, first the field of BO is studied and the mathematical and algorithmic foundation is introduced. The results of this literature study are written down in the section about the theoretical background of the BO (see section 2.3).
Since the goal is to apply BO to the specific algorithm described in [Bosshard et al., 2017] a study of the existing algorithm and code base is needed. For the conciseness of this thesis, it will mostly refer to [Bosshard et al., 2017] and [Bosshard et al., 2018] rather than re-explaining their content. In order to evaluate the possibility of applying the BO to the existing algorithm, empirical experiments are needed. However, the BO algorithm has to be implemented first in a way that it can be combined with the code by [Bosshard et al., 2017]. After the non-satisfactory search for existing BO-code-bases in Java that are concise and modifiable, the decision to write original code for implementing the algorithm was taken. A key benefit from this decision is that the code is directly applied to the existing code by [Bosshard et al., 2017] and does not have to be studied and modified before being integrated.
The implementation is then used for various experiments which are intended to produce results that can be used to explore the possibilities of applying BO to the best-response calculation of the CA-BNE algorithm and to analyse certain attributes of the BO approach.

1.3 Structure of this Thesis

To begin with, chapter 2 will give insight into the topics of combinatorial auctions (section 2.1), algorithmic (section 2.2) and BO (section 2.3). These sections are offering an overview into the general topics with focus on aspects that are important for the course of this thesis. In addition, the most important formal models of this thesis are introduced: LLG-Domain (subsection 2.1.1), core-selecting mechanisms (subsection 2.1.2), Bayes-Nash equilibrium (subsection 2.1.3), Gaussian process (subsection 2.3.1), kernels (subsection 2.3.2) and acquisition functions (subsection 2.3.3).
Chapter 3 walks through the experiments which have been performed to evaluate the possibilities of using the BO as best response calculator. First PS and BO are compared in terms of convergence rate (section 3.1). Then, the focus is shifted towards the variance of the MC integration and how it effects the BO (section 3.2). Lastly, the BO is analysed to see what aspects have to be monitored carefully, when it is applied to the CA-BNE (section 3.3).
In chapter 4, the implementation of the BO algorithm is explained in detail, algorithm definitions are presented and key classes are explained.

2

Problem Overview

In this chapter, an overview of the problem is given together with the formal models. This thesis covers topics from different fields of research like auction theory, (non-)linear optimization and statistics. First, the domain of combinatorial auctions and the concept of BNE are introduced. Secondly, the already existing algorithm by [Bosshard et al., 2017] is explained and it is pointed out how the result of this thesis can help to optimize the algorithm. Thirdly, an illustration of the main optimization method, the BO, is given.

2.1 Combinatorial Auction Setting

In combinatorial auctions (CAs) bidders have preferences over multiple different goods. Since auctions can be considered as allocation problems, CAs map to an interesting set of allocation problems where there are different resources to be allocated efficiently over all the participating bidders [Blumrosen and Nisan, 2007].

Formally, a CA consists of a set of m different and indivisible items and n bidders competing for these items. The bidders have different preferences for the individual items and for bundles/packages of items. For each bidder i the valuation function[1] v_i mapping a valuation to each of the possible bundles of items can be noted. The value of a bundle does not need to be the sum of the individual item's values, it can also be higher (super-additive valuation) or lower (sub-additive valuation). Valid allocations do not assign an item to more than one bidder. An allocation is efficient if it maximizes the social welfare ($\sum_i v_i(S_i)$ where S_i is the bundle of items allocated to bidder i).
The consequence of the super-/sub-additive valuations might become clearer if they are thought of dependencies of individual goods. The value of an item A depends on the allocation of item B. For example, the value a paddle has to you, is much higher if you are also in possession of a boat. Therefore, your valuation for the paddle depends on whether you also have a boat or not. In auctions, bidders announce their value for an item or bundle so that the auction mechanism can allocate the goods among all

[1] These valuation functions respect the concept of free disposal which informally means that adding an unwanted item does not lower the valuation since it can be thrown away without any cost. Formally: $v_i(S) \leq v_i(T)$ if $S \subseteq T$.

bidders in the desired way. However, the value the bidder reports does not need to be the true valuation the bidder has for that good, it can be non-truthful. The reported value $\hat{v}_i = s_i(v_i)$ is called bid and the function s_i that maps the value to a bid, is called *strategy*, with the index i denoting that the focus is on the bidder i. Given all the bids of all bidders, the *mechanism* then decides which bidder gets which bundle of items. It is possible that some bidders do not get any items assigned to them. The mechanism also sets the prices $p(\hat{v}) = (p_1, ..., p_n)$ the bidders have to pay for their allocated items. The actual utility a bidder i receives from the allocated bundle[2] can be expressed by $u_i(v_i, \hat{v}_i, \hat{v}_{-i})$ where v_i is the true, $\hat{v}_i$ the reported value of bidder i and $\hat{v}_{-i}$ the reported values of all the other bidders.
To model the scenario where each bidder only knows its own value and has some expectation about the other bidders' values, bidder valuations are picked randomly from a given distribution. So, the bidders do know their own exact value and the distributions from which the other bidders' values are drawn from.

A combinatorial auction can be held in many different settings, e.g. many different combinations of bidders and items. Dependent on these *domains* the preference profile can be very complex and the analysis of such an auction cost intensive. Such complex domains of auctions often reflect the real world to a great extent and are applied in practice like in spectrum auctions [Cramton, 2013].

2.1.1 LLG-Domain

In order to be able to study the BNE-algorithm and the respective best-response algorithms, they have to be applied to some specific domains. It is favourable to start with rather small domains in which the algorithm does not need an extraordinary time to finish. Such a domain is the *Local-Local-Global* (LLG) domain [Ausubel et al., 2006]. This domain is well-known and used extensively.

In the LLG domain there are two bidders, each of them interested in only one, but not the same, item. Such bidders that are only interested into one item are called *local bidder*. Additionally, there is also one bidder who is interested in the bundle of both items, called *global bidder*. Acquiring only one of these items does not give the global bidder any utility. Next up the exact model of the LLG domain which is used in [Bosshard et al., 2017] and also in this thesis is described.

The valuation is drawn from $\mathcal{U}[0, 2]$[3] for the global bidder. The probability that a local bidder has a valuation $v \in \{0, 1\}$, is v^α $(F(v) = v^\alpha)$, where α is an arbitrary parameter[4]. Since in the LLG domain there are two local bidders it can be that their bids are correlated. This correlation is denoted by γ. The interpretation is that with

[2] denoted by X_i, the bundle of items bidder i is getting
[3] $\mathcal{U}$ denotes the uniform distribution
[4] Note that for $\alpha = 1$ the values for the local bidders are also uniformly distributed, so drawn from $\mathcal{U}[0, 1]$

probability γ both bidders values are equal and with probability $(1-\gamma)$ they are different [Ausubel and Baranov, 2018]. The experiments in this thesis are run with $\alpha \in \{1, 2\}$ and $\gamma \in \{0, 0.5\}$.

2.1.2 Core-Selecting Auctions

As described earlier, the mechanism is what makes an auction select who is getting which items and what prices they have to pay. The most commonly known and used mechanism is the *VCG mechanism* developed by Vickrey(1961), Clarke (1971) and Groves(1973). Despite the preferable property of being truthful, the VCG mechanism has some large shortcomings: With the VCG mechanism, it is possible that the auction generates low or zero revenue, or that bidders do not win the bundle even if they are willing to pay more than the winner does. Increasing the number of bidders should intuitively increase the revenue since higher valuations are more probable and the competition between the bidder is higher, however, the VCG mechanism can result in lower revenue when increasing the number of bidders.[Ausubel and Baranov, 2018] These unpleasant properties originate from the fact that the result produced by the VCG mechanism does not lay in the core (see Definition 1).

Definition 1. *Core: A subset of feasible allocations that are not blocked by any group of bidders coordinating their bids. Allocations in the core are accepted by all bidders individually and even for coalitions of bidders, they would not want to change the outcome. [Ausubel and Milgrom, 2002]*

Because of these shortcomings of the VCG mechanism, other mechanisms were developed which do result in an outcome which lays in the core. These mechanisms are bundled together under the name "core-selecting auctions" or "core-selecting rules". In the following, some core-selecting rules are listed which are based around the LLG domain. For the following notations by [Ausubel and Baranov, 2018] the bid of the global bidder is denoted by b_g and the bids for the local bidders are denoted by b_1 and b_2. The price that the rule assigns, given b_g , b_1 and b_2, is noted as $p(b_g, b_1, b_2)$. For all the rules below, the global bidder always pays the price $b_1 + b_2$ if winning. Thus, the formulas only describe the case where the local bidders win.

Quadratic or Nearest-VCG

The nearest-VCG rule [Day and Cramton, 2012] from now on called *Quadratic* rule [Bosshard et al., 2017], selects the point inside the core that minimizes the Euclidean distance to the VCG outcome. The idea for this originates from [Day and Raghavan, 2007]. A brief intention for this is: Reducing the distance to the VCG result, some of the preferred attributes of the VCG outcome might also apply to the new outcome since they try to be as close together as possible. In addition, the selected outcome has all

the properties of a solution inside the core

$$p(b_g, b_1, b_2) = (0, p_1^V + \Delta, p_2^V + \Delta)$$
$$\text{with } \Delta = \frac{1}{2}(b_g - p_1^V - p_2^V)$$
and where p_1^V and p_2^V are the payments selected by the VCG.

Proxy

The intuition behind the *Proxy* mechanism is that the bidders report their values to a proxy agent who then bids on their behalf in a series of rounds where after each round the agents can adjust the bid. This mechanism was introduced by [Ausubel and Milgrom, 2002]. In the context of the LLG-domain, this mechanism selects that point in the bidder-optimal[5] core which is closest to zero, which means it has the smallest payments. [Ausubel and Baranov, 2018]
Since this is the mechanism that depends the least on the bidder's own bid which leads to the best[6] performance and best incentives as shown by [Ausubel and Baranov, 2018].

$$p(b_g, b_1, b_2) = \begin{cases} (0, \frac{1}{2}b_g, \frac{1}{2}b_g), & \text{if } 0 \leq b_g \leq 2b_2 \\ (0, b_g - b_2, b_2), & \text{if } 2b_2 \leq b_g \leq b_1 + b_2 \end{cases}$$
given that without loss of generality $b_1 \geq b_2$

Nearest-Bid

The *Nearest-Bid* mechanism was introduced by [Ausubel and Baranov, 2018] and its key intention is to select the point which is closest to the winners bid in the bidder-optimal core.

$$p(b_g, b_1, b_2) = \begin{cases} (0, b_g, 0), & \text{if } 0 \leq b_g \leq b_1 - b_2 \\ (0, b_1 - \Delta, b_2 - \Delta), & \text{if } b_1 - b_2 \leq b_g \leq b_1 + b_2 \end{cases}$$
given that without loss of generality $b_1 \geq b_2$ and where $\Delta = \frac{1}{2}(b_1 + b_2 - b_g)$

Proportional

Introduced by [Parkes et al., 2001], the *Proportional* mechanism shares the core constraints b_g[7] among all the local bidders proportional to their bid.

$$p(b_g, b_1, b_2) = (0, b_g \cdot \frac{b_1}{b_1 + b_2}, b_g \cdot \frac{b_2}{b_1 + b_2})$$

[5] Bidder-optimal means that the mechanism chooses an allocation that is optimal for the bidders and not optimal for the seller. For the bidders, the auction is bidder-optimal if the revenue is minimized since then the payments are usually smaller too.

[6] among Proxy, Quadratic and Nearest-Bid

[7] The core constraint for the local bidders is that if they are winning, then together they have to bid at least as much as the global bidder, otherwise the global bidder would be willing to pay more without winning. This would violate the definition of the core.

2.1.3 Bayes-Nash Equilibrium

For the research of auctions it is interesting to find the equilibria, the states where after knowing the outcome, none of the bidders would want to change their bid. Since core-selecting mechanisms are not truthful and therefore truthful bidding is not a dominant and stable strategy, it is needed to actively find the equilibrium. The most important equilibrium definition to consider in this case is the BNE. The BNE is a extension to the broadly known *Nash equilibrium* (NE) introduced by [Nash, 1951]. The NE is defined by a set of strategies for which each strategy yields maximum possible utility given the other strategies in the set. Because in an NE all strategies are best responses to the other strategies, no bidder would want to change its strategy. The BNE was first formally defined by [Harsanyi, 1968] however, for the context of this thesis, an informal explanation is more applicable. In simple words: The BNE is an NE for non-Bayesian games, so games with incomplete information [Harsanyi, 1968, Reeves and Wellman, 2012]. More specific, the bidders do not know the strategies and values of the other bidders, however, they know how these are distributed.

Definition 2. *BNE: A set of strategies so that for each strategy there is no other strategy that yields an expected utility that is strictly greater than the utility the current strategy is producing, given the beliefs about the other strategies. There is no bidder that could improve by changing strategies.*

2.1.4 ε-BNE

The ε-BNE is a slight modification. Instead of saying that each strategy is a strict best response, the ε-BNE says that there is no other strategy that would improve the utility by more than ε, for all strategies [Bosshard et al., 2017].

Definition 3. *ε-BNE: A set of strategies so that for each strategy there is no other strategy that yields an expected utility that is greater by more than ε compared to the utility the current strategy is producing, given the beliefs about the other strategies. There is no bidder that could improve by more than ε by changing strategies.*

2.2 Algorithmic Setting and Optimization

In this section, the CA-BNE algorithm of [Bosshard et al., 2017] is explained. This section should give an overview of the different parts of the algorithm and how they work, especially focusing on the best response calculation.

2.2.1 Two Phase Solution

The algorithm by [Bosshard et al., 2017] does separate the finding of the BNE into *the search phase* from the estimation of the ε for the found BNE and *the verification phase*. This separation of phases if one aspect in which the work of [Bosshard et al., 2017] differs from prior work.

Search Phase

In the search phase, the BNE is calculated using an iterative process. The algorithm starts with a preset strategy, for example, truthful bidding where each bidder bids its value. Then for each bidder, the algorithm calculates the best response for all possible valuations, given the strategies of the other bidders. This results in a new strategy for that bidder. This strategy then takes the place of the initial strategy and a new iteration starts.

The best response is calculated by maximizing the utility function for a given valuation as well as for the strategies of the other bidders. Since each bidder does not know the exact valuations of the others bidders it is only possible to maximize the utility in expectations. Therefore, the algorithm draws valuations of other bidders according to the known distribution and then maps them to bids using the strategy resulted from the last iteration. From all these bids, together with a bid b_i of the bidder in focus, the outcome can be calculated applying the mechanism. The utility can be derived out of the initial value v of that bidder and the outcome. To get a representative result, this drawing of valuation for the other bidders has to be done many times. This process is often called *sampling*. In the end, the results of all samples are combined and a mean is computed. This mean is then the expected utility for that bidder when bidding b_i and having value v. The combined process of sampling and combining the result to a single "in expectation" utility-value is done by the MC integration.

To get the best response, the maximal expected utility has to be calculated over all possible bids (still given one explicit value). In the original CA-BNE by [Bosshard et al., 2017] PS is applied to sequentially evaluate the expected utility for some bids (see [Bosshard et al., 2017] for information on how the PS selects the next points to evaluate). The PS results in finding the one bid that yields the highest expected utility - the best response.

Having found the best response, for one given value, the next step is to find the best response for all possible values, so that finally a strategy can be defined. In practice, the algorithm calculates the best response for some, but not all values and then fits a function through these points to construct the strategy.

Verification Phase

The work of this thesis does not affect the verification phase at all, therefore see [Bosshard et al., 2017] for more information about the reasoning and the process behind the verification step. Just as a short summary: The verification phase is needed to calculate the ε which then decides if a good enough equilibrium is found or if the search has to be continued.

2.2.2 Full Action Space

Another aspect in which the work of [Bosshard et al., 2017] differs from most others is that the full action space is considered. This means the strategies have access to bids and values from a continuous space. This stands in contrast to prior algorithms that restrict their action space to keep the runtime of the algorithm manageable. For the course of this thesis, it is just important to keep in mind that the modified best response calculation still has to work on the continuous space. For the detailed reason and consequences of considering the continuous space see [Bosshard et al., 2017].

2.3 Bayesian Optimization

The BO methods got much attention in the past years with the vast development of machine learning. It is very useful for finding extrema of objective functions of which no concrete expression has been found. BO methods are especially powerful if the objective function is expensive to evaluate. It has been shown that BO is one of the methods which requires the least evaluations [Jones et al., 1998]. Therefore if the time to evaluate the objective function at one point is much larger than the overhead that the BO method produces, then BO increases the performance substantially. The name "Bayesian optimization" originates from the fact that this method incorporated the "Bayes' theorem" which is used to argue in terms of beliefs about the probability of different objective functions and how they are fitting the already made observations. These beliefs are also called *prior*.

Theorem 1 (Bayes' theorem)**.**

$$P(M \mid E) \propto P(E \mid M)P(M)$$

The probability that a model M explains the given observations E is proportional to the probability of observing E given the model M multiplied by the probability of the model itself.

Let x_i be the i-th sample and $f(x_i)$ the respective observation of the objective function. Given some prior belief, it can be estimated how likely different functions fit the given observations. In general, the prior belief is that the objective function is rather smooth and noise free (this belief is not mandatory, BO can be adjusted to fulfil this belief). This results in an estimate of the mean and related variance range of the objective function. Given mean and variance, a so-called *acquisition function* calculates which point increases the knowledge about the objective function the most. This point shall be evaluated next. Depending on which acquisition function is applied, the BO can be steered towards the exploration of the whole space or towards the exploitation of a specific area where the maximum might lay.

2.3.1 Prior over Objective Function - Gaussian Process

The Gaussian process (GP) can intuitively be explained as the equivalent to the Gaussian distribution which assigns probability to functions instead of values. So, the GP turns a Gaussian distribution with multiple variables into a infinite-dimensional stochastic process [Brochu et al., 2010]. The distribution defined by the GP is fully specified by the mean μ and covariance function k.

$$f(x_j) \sim \mathcal{GP}(\mu(x_j), k(x_j, x_l)) \tag{2.1}$$

with $\mu(x_j) = \mathbb{E}[f(x_j)]$ and $k(x_j, x_l) = \mathbb{E}[(f(x_j) - \mu(x_j))(f(x_l) - \mu(x_l))]$ [Rasmussen and Williams, 2006]. The variables x_j and x_l are two input values which correspond to two possible bid values in the LLG context. Equation 2.1 can be rewritten in a matrix formulation.

$$\mathbf{y} \sim \mathcal{N}(\mu, \mathbf{K}) \tag{2.2}$$

with $\mathbf{K}$ being the kernel matrix defined by $\mathbf{K}_{jl} = k(x_j, x_l)$ and $\mathcal{N}$ being the multivariate normal distribution.
The model assumes that the prior mean μ is zero. Extending the notation to multiple input values, then the input values (bids) of the observations are denoted as a vector $\mathbf{x}$ and the target values (utility) as vector $\mathbf{y}$. When the goal is to evaluate the prior at a test-point with bid-value x_* to get the expected utility y_*, then equation 2.2 can be extended the following way:

$$\begin{bmatrix} \mathbf{y} \\ y_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(\mathbf{x}, \mathbf{x}) & K(\mathbf{x}, x_*) \\ K(x_*, \mathbf{x}) & K(x_*, x_*) \end{bmatrix}\right) \tag{2.3}$$

[Rasmussen and Williams, 2006]. The matrix inside the squared brackets denotes the extended covariance matrix. For equation 2.3, the conditional distribution can be derived (see [Rasmussen and Williams, 2006] A.2)

$$y_* \mid \mathbf{y} \sim \mathcal{N}\big(K(x_*, \mathbf{x})K(\mathbf{x}, \mathbf{x})^{-1}\mathbf{y}, K(x_*, x_*) - K(x_*, \mathbf{x})K(\mathbf{x}, \mathbf{x})^{-1}K(\mathbf{x}, x_*)\big) \tag{2.4}$$

This is equal to the predictive distribution $P(y_* \mid \mathbf{y}) = \mathcal{N}(\mu(x_*), \sigma^2(x_*))$ [Brochu et al., 2010]. As a result, the mean μ and the variance σ^2 of an additional target value y_* at x_* can be expressed in form of previous observations ($\mathbf{y} = f(\mathbf{x})$) as follows:

$$\mu(x_*) = K(x_*, \mathbf{x})K(\mathbf{x}, \mathbf{x})^{-1}\mathbf{y} \tag{2.5}$$

$$\sigma^2(x_*) = K(x_*, x_*) - K(x_*, \mathbf{x})K(\mathbf{x}, \mathbf{x})^{-1}K(\mathbf{x}, x_*) \tag{2.6}$$

Noise

In reality, the observations are often covered in noise, in that case the prior changes slightly. The noise changes $\mathbf{y} = f(x)$ into $\mathbf{y} = f(x) + \varepsilon$ where ε is the noise of the n observations. Starting from the premise that the noise has a normal distribution $\varepsilon \sim \mathcal{N}(\mathbf{0}, {\sigma_n}^2)$, and therefore is so called Gaussian noise, the variance of the noise can

be added to the variances of the observations. The equation 2.3 changes to the following [Rasmussen and Williams, 2006].

$$\begin{bmatrix} \mathbf{y} \\ y_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(\mathbf{x},\mathbf{x}) + {\sigma_n}^2 I & K(\mathbf{x}, x_*) \\ K(x_*, \mathbf{x}) & K(x_*, x_*) \end{bmatrix}\right) \tag{2.7}$$

Further, 2.5 and 2.6 change into:

$$\mu(x_*) = K(x_*, \mathbf{x})[K(\mathbf{x},\mathbf{x}) + {\sigma_n}^2 I]^{-1}\mathbf{y} \tag{2.8}$$

$$\sigma^2(x_*) = K(x_*, x_*) - K(x_*, \mathbf{x})[K(\mathbf{x},\mathbf{x}) + {\sigma_n}^2 I]^{-1}K(\mathbf{x}, x_*) \tag{2.9}$$

2.3.2 Covariance Functions - Kernels

The covariance function k, also called *kernel*, is a key part of the GP and thus also of the BO. In the kernel, one can encode the assumptions and beliefs that are made about the function that is to optimize. Depending on the choice of the kernel properties like smoothness can be presumed.

Squared Exponential Kernel

Probably the most common kernel is the Squared Exponential (SE) Kernel [Rasmussen and Williams, 2006], also referred to as Gaussian Kernel:

$$k_{SE}(x_i, x_j) = exp\Big(-\frac{1}{2\theta^2}\|x_i - x_j\|^2\Big) \tag{2.10}$$

This function is infinitely differentiable because it is a exponential function. Therefore the resulting objective function is very smooth. The *hyperparameter* θ is often called length-scale parameter. Its effect can be best described in the one-dimensional case where it acts as a length scale [Rasmussen and Williams, 2006]. The smaller θ, the smaller is the interval from which evaluated points are considered to make the predictions. Therefore, when using a small θ, the prediction can oscillate rather fast and the variance is rather high. The SE kernel enjoys great popularity since it is simple and only adds one hyperparameter.

Matern Kernel

If the goal is to have more flexibility by adjusting the smoothness of the modelled function the Matern Kernel [Matérn, 2013, Stein, 2012] can be used. This kernel is more complex but also adds only one hyperparameter ζ, the smoothness-parameter. The kernel looks like the following:

$$k_{Matern}(x_i, x_j) = \frac{1}{2^{\zeta-1}\Gamma(\zeta)}(2\sqrt{\zeta}\|x_i - x_j\|)^{\zeta} H_\zeta(2\sqrt{\zeta}\|x_i - x_j\|) \tag{2.11}$$

where Γ is the Gamma function and H_ζ is the ζ-order Bessel function [Brochu et al., 2010].

2.3.3 Guide the Search - Acquisition Functions

The GP returns mean and variance for the given points. Now the BO has to define where it wants to evaluate the objective function next. That is where the acquisition function comes into play. It defines the strategy of how to determine which point yields most improvement when evaluating. Usually, the acquisition function makes a trade-off between points where the mean is high and ones where the variance is low.

Various different acquisition functions have been developed. The three most common ones are presented in the following.

Probability of Improvement

One of the most basic acquisition functions is the *probability of improvement*. This acquisition function was introduced by [Kushner, 1964]. It is purely based on getting a point with higher expected value, in our case utility, than the highest function-value of all evaluations so far. The evaluation of the objective function with the current function-value is called *incumbent* and is defined by $f(x^+)$ where:

$$x^+ = \underset{x_i \in \mathbf{x}}{\operatorname{argmax}} f(x_i) \tag{2.12}$$

[Brochu et al., 2010]. However, this acquisition function does completely neglect the extent of the improvement. This acquisition function can end up searching around a local maximum (exploiting) and thus neglect the exploration missing a global maximum. It is defined as:

$$PI(x) = P(f(x) \geq f(x^+)) = \Phi\left(\frac{\mu(x) - f(x^+)}{\sigma(x)}\right) \tag{2.13}$$

With $\mathbf{\Phi}$ being the normal cumulative distribution function (CDF).

Probability of Improvement With Trade-off

To favour points with greater improvements, a trade-off parameter ξ can be introduced [Kushner, 1964, Brochu et al., 2010]. This parameter specifies how much higher an evaluation of a candidate must be in comparison to the incumbent. This way, the chance of finding the global maximum is higher. The formula changes as follows:

$$PI_+(x) = P(f(x) \geq f(x^+) + \xi) = \Phi\left(\frac{\mu(x) - f(x^+) - \xi}{\sigma(x)}\right) \tag{2.14}$$

The parameter ξ can be static but also dynamic, when it is high then the algorithm is encouraged to explore. As ξ goes to zero, PI_+ goes to PI.

Expected Improvement

The Expected Improvement function is even more advanced since, in addition to the probability, it also considers the magnitude of the possible improvement. The analytical form derived by [Mockus, 1975] looks as follows:

$$EI(x) = (\mu(x) - f(x^+))\Phi(Z) + \sigma(x)\phi(Z) \tag{2.15}$$

where ϕ is the probability density function (PDF), Φ the CDF and Z is a variable defined by:

$$Z = \frac{\mu(x) - f(x^+)}{\sigma(x)} \tag{2.16}$$

3

Experiments

In this chapter, we discuss three experiments that should give us further insights into how the BO performs in the setting of the CA-BNE algorithm. For the following experiments we use the same 16 domains which are also considered in [Bosshard et al., 2017], these are the Proxy, Proportional, Quadratic and Nearest-Bid domains with each of them having four sub-domains with $\alpha \in \{1, 2\}$ and $\gamma \in \{0, 0.5\}$.

3.1 Convergence Comparison

In this experiment, the BO approach is compared to the PS approach in terms of convergence. The goal is to see whether it is possible to find the optimum faster when using BO. We will also define what metric is the subject of measurement and the convergence speed is quantified.

The main object of research in this thesis is the best response algorithm as part of the CA-BNE algorithm developed by [Bosshard et al., 2017]. This sub-algorithm is thought to have a great potential for increasing the overall efficiency since it contains the usually costly MC integration which is used for the evaluation of the utility function. If we are able to reduce the number of these evaluations, then the MC integration is not executed that often and the runtime might be reduced. Since the goal is to improve the performance of the CA-BNE algorithm, we rather look at the best response algorithm within the scope of the CA-BNE and not isolated. Additionally, we are able to notice effects on calculations outside of the best response algorithm when looking at the whole algorithm. Such an effect could, for example, be that strategies produced by the BO might oscillate around the true equilibrium-strategy and thus the ε will never be small enough to make the CA-BNE algorithm stop.[1]
The main difficulty when comparing both algorithms (BO and PS) is that for the PS algorithm, which is used by [Bosshard et al., 2017], the parameters are already fairly well configured. The configuration is done in a way that PS does not have a convergence-based stopping criterion but rather it has a hard limit on how many iterations it should

[1] Only when ε is smaller than a given threshold (here 10^{-5}) the CA-BNE algorithm stops searching. See [Bosshard et al., 2018] for more information on the stopping criteria

do. As a result, if the BO uses a self-determined convergence-based stopping criterion, it might run longer but calculate the result more exact. However, if it uses a fixed iteration limit, the runtime is just directly related to that limit. Either way, we won't get a fair comparison nor a reliable estimation of whether the BO does indeed use fewer evaluations to generate a result of comparable quality.
Besides that, only measuring the runtime of the whole CA-BNE algorithm could also provide false information, since it might happen, that the two subroutines provide slightly different results. Depending on the accuracy of the found best responses, the ε can be different for BO and PS and thus, the CA-BNE might have to calculate more strategies (more BNE-iteration) until it converges. In other words; it can happen that it needs more iterations to converge, which leads to a longer runtime of the whole BNE algorithm even though the best response is found faster.

3.1.1 Experimental Setup

The above-noted reasons lead to an experiment where the main focus lays on the best response calculation. Both algorithms, PS and BO, are configured in a way that by default they have a fairly large maximum iterations limit. However, both algorithms are stopped early if they reach a result reasonably close to the ground truth optimum. This ground truth optimum has to be determined first. For that, we evaluate the utility function on a grid and search for the maximum using a normal grid search. We will discuss later, in more detail, how the ground-truth-calculation works.
We note how many iterations and how many evaluations the algorithms needs, until they get stopped. The number of evaluations defines how many MC samples have to be drawn. We do not run the best response calculation isolated because we want to observe if the changes to the best response calculation have any effect on the rest of the BNE algorithm. We will run this experiment twice for each domain, once we use the PS as the algorithm which determines the next strategy for the BNE algorithm and the other time we use the BO. This way, we might also notice any effects if we use a best response calculation that uses strategies produced by a different best response calculation.

Common Random Variables

To achieve comparable results even without extremely high sample-sizes, the `CommonRandomGenerator` (see 4.6) is used. This generator ensures that both algorithms are run with the same set of random samples. Therefore, the results can be directly compared without the need of including any MC integration based variance in the analysis. Using the same samples means that the utility functions in both algorithms looks exactly the same and that they have the global optimum at the same point because the drawn valuations for the bidders are identical.

Ground Truth Calculation

To be able to calculate a ground truth in a reasonable time, we restrict ourselves to a grid, meaning we only allow bids that lay on a predefined grid to be used. For the domains considered in this experiment, the grid is only one dimensional which means the bids are taken equidistant and uniformly from the bid range of 0 to the maximal valuation of the bidder. To get the optimum we then take the maximal utility of all these points.
To make sure that we do not end up in different optima PS and BO are modified to only consider bids from this same grid and no values in between grid points.

For the PS we round all bids that are used during the search to the nearest point on the grid and use these grid points for the further steps and evaluations. In the BO we chose to use a different acquisition function optimization algorithm since with the default, rather efficient, Brent search there is no way to enforce grid points. The only way to achieve this would have been to round the points post-search, which appears to be a inferior way of doing it since it could happen that a rounded point close to the found optima is worse than another point on the grid. Therefore we exchanged the Brent search with a grid search that optimizes the acquisition function on this same grid. As a result, the bid which maximizes the acquisition function and is evaluated next, is also on the same grid.

3.1.2 Configuration

For the experiment we set the maximum number of iterations to 25, for both the PS and the BO.[2] For the surrounding parts of the BNE algorithm we keep the configuration at default. This means ε is set to 10^{-5} and the amount of BNE-iteration is limited to 30 iterations. Regarding the grid of values to calculate the best response for, we use a grid with 40 points. The whole experiment is done with a sample size of 10000 samples per MC integration, for the ground truth as well as the PS and BO. For the ground truth and the optimization algorithms, we take the bid from a grid with 1000 uniformly distributed points. The selected number of 1000 points for the grid is also a compromise between accuracy and reasonable runtime. All optimizers are run with exactly the same samples. The number of grid points is not of great importance since our goal is not to be as exact as possible but more to get the same optimum when using the same samples. The confidence range is set to 10^{-5}, so the optimizers have to find a point with expected utility less than 10^{-5} away from the ground truth optima.

Pattern Search

For PS we stay consistent with the default configuration of `Pattern-Size` of 3 and with `Stepsize` 0.1. The starting point is still the `currentBest` bid from the previous strategy.

[2] Even though PS does up to two evaluations per iteration and BO only one it is still viable to use an iteration based limit since the limit is hardly ever reached.

For the 4x4 domains we are testing here, we use the `UnivariatePattern`.

Bayesian Optimization

To start the BO, we need initial training points. We use the `currentBest` from the previous strategy and two (quasi[3]-) random points for that. For the global bidder the length-scale parameter is set to 0.6 and 1.0 for the local one[4]. The length-scale parameters are chosen rather large because we assume smooth utility functions. We use the slightly more advanced version of BO where we include the noise of the MC integration into the definition of the covariance matrix. The variance of the noise of the MC integration is calculated using the `expensiveVariance` module.

3.1.3 Results

First, we will look at how the number of needed evaluations evolves over the different BNE-iterations. We see that especially for the first few BNE-iteration, BO needs much fewer evaluations, compared to PS. Figure 3.1 shows the needed evaluation for each iteration as a graph, comparing BO and PS for all four subdomains of the Nearest-Bid mechanism. The Figure 3.1 also indicates that there are only small differences between the different sub-domains. This is the case for all domains except the Proxy, where the needed evaluations are not consistent over the different α and γ and BO needs more evaluations compared to PS as shown in Figure 3.2. For both Figures 3.1 and 3.2 we used the strategies found by the PS. There is no evidence that there is a difference between the experiments which use PS as strategy-defining methods and the experiments which use BO.[5]

Figure 3.1, 3.2 and A.1-A.8 present some interesting results, however, they do only represent a whole iteration and how the needed number of evaluations evolves over multiple iterations. From these results it is not possible to see how different bidder's valuations impact the number of needed evaluations. Therefore, we now also look into some valuation based results with some other key values that can be considered as important for the course of the thesis. We want to find out how many evaluations[6] the different optimization strategies (BO and PS) need at most. The maximum number of evaluations needed for any single valuation over the whole search phase are listed in the Tables 3.1 and 3.2 for all 4x4 domains. In addition, we also noted down the maximum median per iteration. This means we calculate the median over all valuations, for each

[3] Uniformly distributed points on the bid-space to cover the whole range rather only on a part which would be possible by true randomness

[4] These values have been found to perform reasonably well by doing some empirical testing, however, they are not proven to be the best and with extensive test better parameter would be found

[5] The plots for the same domain but once BO based and once PS based, should not be directly compared as they do not share the same random variables, hence differences can naturally appear.

[6] Note that the iteration limit is set to 25 iterations, but we are looking at evaluations here, therefore we can get at most 28 evaluations for the BO (one for each of the 25 iterations and 3 initial trainingspoints). For PS, we could theoretically get up to 50 evaluations. However for both algorithms (BO and PS) the limit is barely ever reached.

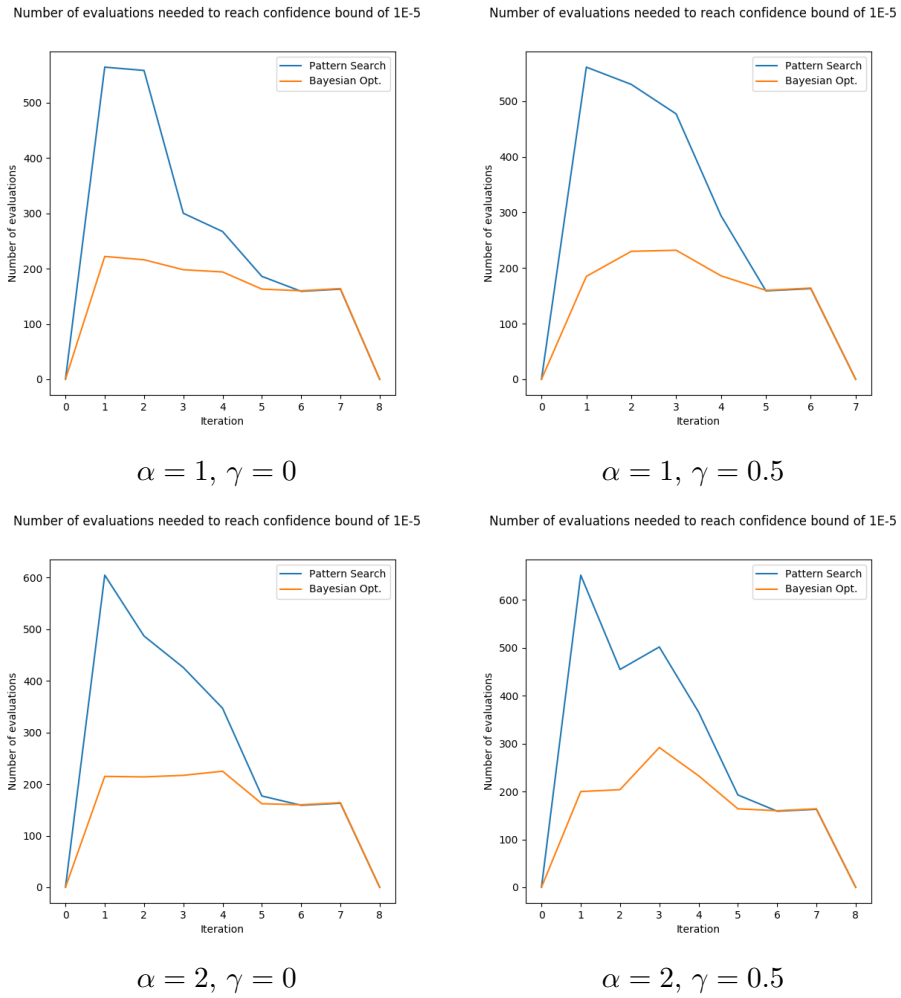


Figure 3.1: **Needed evaluations for Nearest-Bid mechanism, PS is strategy-defining.** Convergence for the Nearest-Bid mechanism, comparing how many evaluations are needed to find a point in a region of 10^{-5} around the ground truth optimum. The number of evaluations are summed over all 40 valuations that are taken for the local bidder. Strategies used for the next iterations are based on the PS. The blue solid line represents the number of evaluations needed for the PS, orange for the BO.

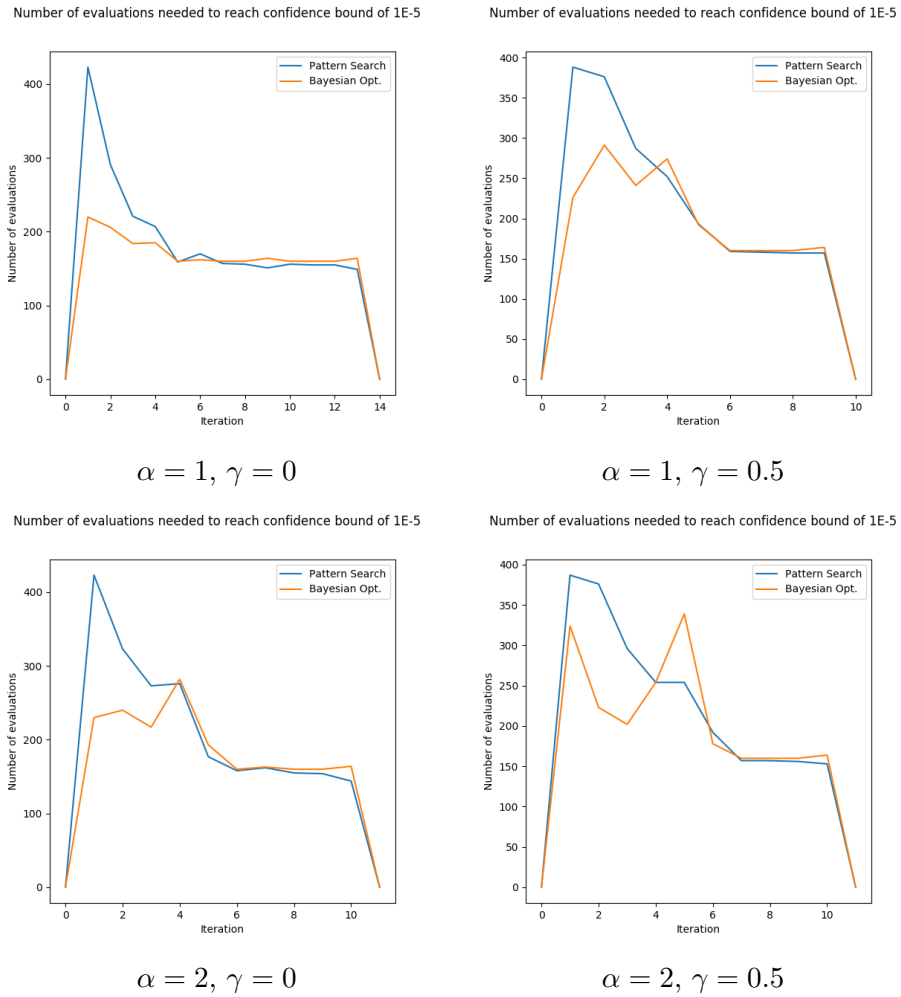


Figure 3.2: **Needed evaluations for Proxy mechanism, PS is strategy-defining.** Convergence for the Proxy mechanism, comparing how many evaluations are needed to find a point in a region of 10^{-5} around the ground truth optimum. The number of evaluations are summed over all 40 valuations that are taken for the local bidder. Strategies used for the next iterations are based on the PS. The blue solid line represents the number of evaluations needed for the PS, orange for the BO.

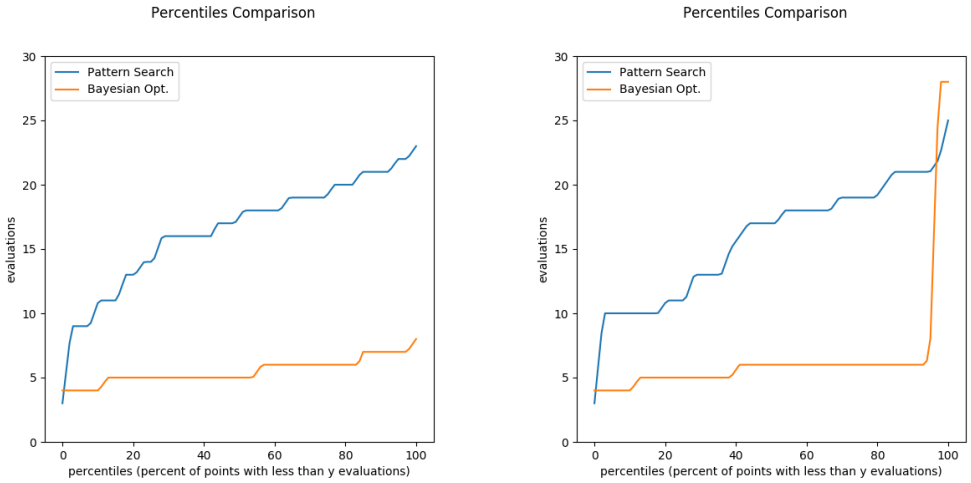
iteration, and then choose the maximum over all iterations. As it is visible in Tables 3.1 and 3.2, the maximum number of evaluations can vary a lot between optimization approach, domain and sub-domain. More convicting is the maximum median over the iterations. The reason for the maximum not being very meaningful is that it can happen that for one single point, the optimization needs many evaluations but only a few at all the other points. Having the median as side metric, we see how well the optimization performs in general. The maximal median is noted in brackets. We see that for BO, the maximal median is, consistently over all of the 16 domains, substantially lower than for PS.

The median indicates that for BO half of the points need 5 to 7 evaluations or less,

mechanism	optimizer	α=1/γ=0		α=1/γ=0.5		α=2/γ=0		α=2/γ=0.5	
NearestBid	PS	23	(17)	25	(17)	23	(18)	25	(18.5)
	BO	8	(5)	28	(6)	28	(6)	16	(6)
Proportional	PS	19	(11)	19	(13)	20	(17)	22	(18)
	BO	7	(5)	7	(5)	11	(5)	10	(6.5)
Proxy	PS	17	(11.5)	17	(10)	17	(11)	17	(10)
	BO	12	(5)	28	(5.5)	28	(5)	28	(6.5)
Quadratic	PS	19	(11)	21	(16)	21	(17)	20	(16)
	BO	28	(5)	28	(5)	9	(6)	8	(5)

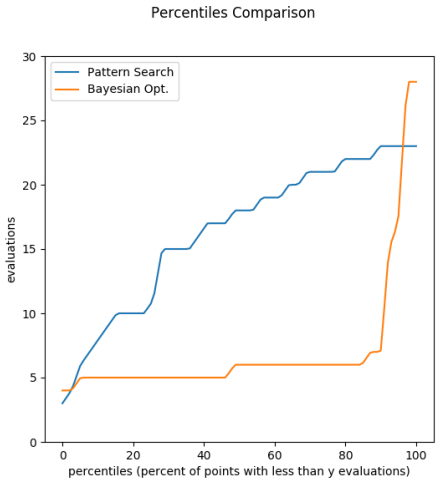
Table 3.1: **Overview of maximum and median of needed evaluations for all domains, BO is strategy-defining.** This table shows the maximum number of evaluations that where needed to find the optima (bid) for a single point (valuation). The number in brackets denotes the median. Note that the median is always taken over one iteration and the maximum of all these medians is written in this table. For this table results of experiments where the BO was used to define the next strategies, are used.

whereas for PS half of the points need 11 to 15 evaluations or less. As interesting as this result may be, it just states that for 50% of the values the BO perform better. However, instead of looking just at the median, we can look at possible percentages (percentiles). When doing this, we see that BO, in general, performs better on up to 80% of the points, as you can see in Figure 3.3, which are the results for the Nearest-Bid mechanism and which represents the result of most domains fairly well. The result for all domains can be found in the Appendix (Figure A.9 - Figure A.16). In these plots, we calculated all percentiles for all possible valuations. A p-percentile denotes how big a value c has to be until p% of all the results are lower than c. We then take the maximum over all the iterations. The result expresses a quasi-worst-case situation. We can also see that, mostly, only the last few percentiles have a large increase in needed evaluations. The only main domain that does look a bit different is the Proxy where the BO is strategy defining and $\gamma = 0.5$ (Figure: 3.4): Here, BO has a larger amount of points that need more evaluations than PS.

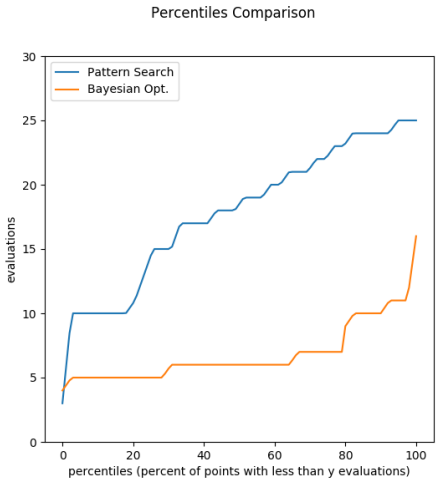


$\alpha = 1, \gamma = 0$

$\alpha = 1, \gamma = 0.5$

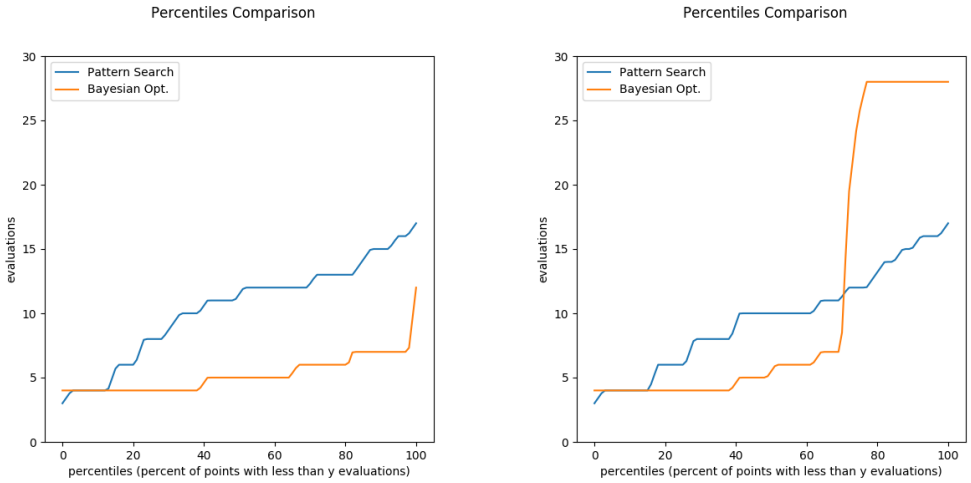


$\alpha = 2, \gamma = 0$



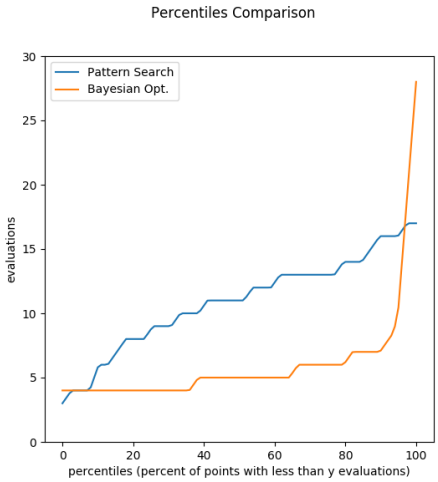
$\alpha = 2, \gamma = 0.5$

Figure 3.3: **Percentiles for the Nearest-Bid mechanism, BO is strategy-defining.** This graphs show all different percentiles for the PS and the BO for the Nearest-Bid mechanism. On the y-axis is the number of evaluations which are required so that x% (x-axis) of all points converge to the ground truth optimum. For example for the PS in the domain with $\alpha = 1$ and $\gamma = 0$, 80% of the points need 20 or less evaluations to converge, for the BO 80% only need 6 evaluations or less.

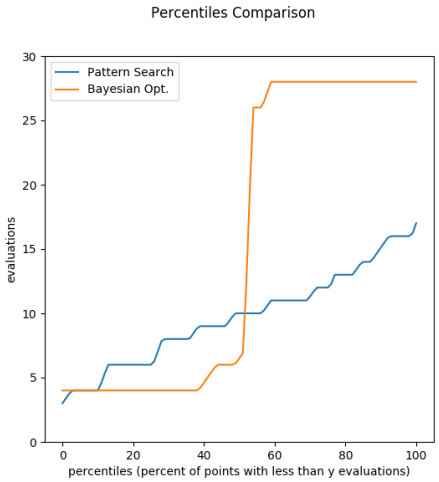


$\alpha = 1, \gamma = 0$

$\alpha = 1, \gamma = 0.5$



$\alpha = 2, \gamma = 0$



$\alpha = 2, \gamma = 0.5$

Figure 3.4: **Percentiles for the Proxy mechanism, BO is strategy-defining.** This graphs show all different percentiles for the PS and the BO for the Proxy mechanism. On the y-axis is the number of evaluations which are required so that x% (x-axis) of all points converge to the ground truth optimum. For example for the PS in the domain with $\alpha = 1$ and $\gamma = 0$, 80% of the points need 14 or less evaluations to converge, for the BO 80% only need 6 evaluations or less.

mechanism	optimizer	α=1/γ=0		α=1/γ=0.5		α=2/γ=0		α=2/γ=0.5	
NearestBid	PS	22	(15)	25	(15)	23	(16.5)	23	(18)
	BO	25	(5)	7	(6)	24	(6)	20	(6)
Proportional	PS	19	(11)	19	(13)	20	(17)	22	(17)
	BO	8	(5)	7	(6)	13	(6)	10	(6)
Proxy	PS	17	(11.5)	16	(11)	17	(11.5)	22	(10)
	BO	8	(6)	28	(6)	28	(5.5)	28	(6.5)
Quadratic	PS	19	(11)	19	(12)	20	(17)	22	(17)
	BO	7	(5)	7	(6)	12	(6)	8	(6)

Table 3.2: **Overview of maximum and median of needed evaluations for all domains, PS is strategy-defining.** This table shows the maximum number of evaluations that where needed to find the optima (bid) for a single point (valuation). The number in brackets denotes the median. Note that the median is always taken over one iteration and the maximum of all these medians is written in this table. For this table results of experiments where the PS was used to define the next strategies, are used.

3.1.4 Conclusions

The results presented in Tables 3.1 and 3.2 point toward the conclusion that using BO does indeed reduce the number of iterations that are needed to reach a approximative region around the true optimum. We can also see that even though BO has the trend of lowering the number of evaluations it often happens that there are some points where finding of the optimum needs the same number or even more evaluations as PS. As a result of this, it is not possible to lower the iteration limit without risking getting a wrong optimum in some of the points. However, if some inaccurate results do not matter, it is possible to lower the limit when using BO. Together with Figures 3.1 and 3.2 we can draw the conclusion that especially in the first few iterations of the BNE algorithm, the difference between PS and BO is huge. Here, BO can cut the number of evaluations needed in halves. It makes sense that BO surpasses PS in the first iterations since we start with a strategy that might be far away from the equilibria and hence the strategies can change a lot from one iteration to the next one. This means the optimization approach has to search a wide range of possible bids. Since BO is constructed in a way that allows it to select a bid from the whole action space for the next evaluation, it can easily and with just a few evaluations explore the whole region. The PS on the other hand always has to start at the starting point (best result from the last iteration) and has a fixed step size assigned to it. When exploring the region it can only make steps as big as the step size. As a result, it might take the PS multiple steps to get sufficiently far away from the starting point towards the optimum. But in the last iterations, BO and PS perform almost equally. The interpretation for this result is fairly simple. Since we are searching for an equilibrium, the strategies do change less towards the end of the CA-BNE algorithm. Additionally, PS starts at the optimum from the last

iteration and as the strategies do not change a lot, PS starts already close to the true optimum. The BO also uses this previous maximum as one of the training points, hence both approaches, BO and PS, start with a point close to the optimum and therefore have no need to search for an extended period of time. The reason why BO sometimes performs worse than PS is that BO is not bound to the starting point and is freer to explore the complete space. We can even back up these result with the analysis of the percentiles which show that for most domains there are only very few points that need more evaluations in BO. This means that for 80% of all points, the BO needs fewer evaluations. Among the remaining points, examples are found that take a considerable amount of more evaluations. This can originate from the fact that it is possible that BO sometimes can get a bad convergence towards the optimum because it is testing for the next evaluations points very close to the previously evaluated values. This can very well happen if the function which should be optimized is flat.

3.2 Variance Exploration

In this experiment, the variance of the evaluation of the utility-function is explored. For the BO it is important to know how big the variance of the evaluation of the utility function is. Especially when using a BO implementation that takes noise into account, as we do. Therefore, it is useful to explore and understand how the variance of the setting behaves for the different (sub-)domains.

To calculate the utility for a fix valuation and a given bid, the algorithm uses the MC integration. Hence, it draws random[7] bids according to the given distribution for the other bidders and then applies the mechanism to calculate the outcome and the utility. To get a reliable result, it draws many samples and then averages the intermediate results.

Knowing how the variance behaves might be useful to construct more efficient acquisition functions or kernels. Also when using the BO, which includes the noise, the variance is needed. However, calculating the variance is rather expensive, so if we can approximate the variance by a general model, we can save computational time. Also we check whether the accuracy of the noise has an impact on the performance of the BO.

3.2.1 Experimental Setup

Since the variance depends on the combination of valuation and bid, we calculate the variance on a grid. The variance can depend on the strategy, therefore these grid evaluations are repeated for all strategies.
In the context of MC integration, there are multiple understandings of variance, so it

[7] Theoretically, the values should be i.i.d. (independent, identically distributed) However, the CA-BNE algorithm makes use of some tricks that enables the algorithm to perform better even without i.i.d. random bids

is important to be clear about which interpretation of variance we are using in this experiment. Often, in MC integration, the "in-sample" variance is used. This variance definition describes how far away the result of a single evaluation is from the mean over the whole group of samples. However, this is not what we are interested in. We want to know how far away the mean of the whole sample group is from the correct result. And thus we face our first problem; There is no analytical representation for the real result. We decided to evaluate the utility with a very large sample-group to have a mean that should be representative. This result we call benchmark.
Also important for getting a reasonable calculation of the variance is, that the MC integration uses true random variables. If there is any correlation between the samples then the variance calculation is biased and the result is not representative. To make sure that this is not the case we use the `NaiveRandomGenerator` to generate the samples and not the `CommonRandomGenerator` as in the normal use-case. In the `NaiveRandomGenerator` every sample is drawn independently and no cache is created. We do that for the calculation of the benchmark as well as for the normal result with fewer samples. However, calculating the actual variance only once is not enough. This would only represent the variance of a single sample set. What we want is a general variance result. Therefore, we repeat the process multiple times. This means we take multiple sample sets and compare the result to the benchmark value. We then average all these variances to the final variance. To get the variances in a most fitting environment we run the CA-BNE algorithm with a PS around it and use these strategies for the variance calculation.

3.2.2 Configuration

In this experiment, we use 100'000 samples to get the benchmark result. As the actual sample size for the experiment, we use 10'000 samples as it is also the default `SampleSize` and seems to be of reasonable size. We take 100 sub-samples for which we calculate the variance and then average over them.
To keep the runtime manageable, we use a 40 x 40 grid. So we take 40 uniformly distributed valuations from the original (given by LLG) value-range as well as 40 uniformly distributed bids from the bid range.
For the surrounding PS, the usual configuration is chosen. The only exception is that the grid is also fixed to 40 grid points.

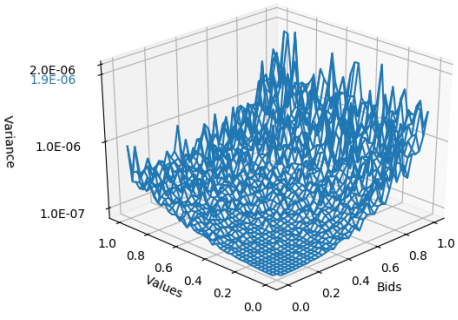
3.2.3 Results

Looking at the different plots (Figures 3.5 - 3.8) of the different domains, it becomes clear that the variance behaves very similar for all the different domains and sub-domains. We are only considering the overall shape and ignore the spikes[8]. For all of them, the variance increases with the bid as well as with the value. This is consistent with the theory since for low bids the bidder will lose most of the time and get an utility of 0.

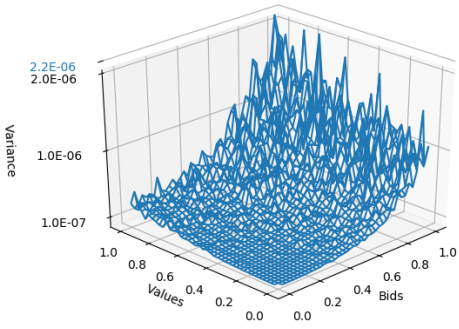
[8] In general we would expect the shaped to be rather smooth, however since the variance is rather small it can happen that in our setup for some sample-sets the variance is far away from the other which then has an impact on the average.

The rare cases that the bidder wins the utility[9] depends purely on the valuation. For low valuations, the bid and the valuation are close and therefore the utility rather small. For large valuation and small bids, it is possible that the price, that has to be paid is small, which then results in a rather large utility. This explanation is very general, however, due to this generality, it can be applied to all the different domains. When looking closer at the different domains we can see a difference between domains with $\gamma = 0$ and $\gamma = 0.5$. We see that for domains with $\gamma = 0.5$, where the local bidders' values are correlated with a probability of 0.5, the dependency on the value is smaller. This means that for the same bid, the increase of the variance with increasing value is less compared to the domain with $\gamma = 0$. This can be explained by the fact, that given a valuation for one local bidder the probability that the other local bidder's value is from a similar magnitude, is high. Thus the variance in value is smaller and therefore, also the variance in utility. In addition, when one local bidder has a low value, then the other has, with the probability of 0.5, a low valuation too, which means that the global bidder does not need to bid much to get both items. Hence the chances of winning the item are lower for the local bidder. It should be clear that for not winning the item the variance is low anyway since the utility is always 0. Thus the overall variance is lowered. We can also see for $\alpha = 2$, that the variance has the shape of a parabola over the values for maximal bids. Explicitly for a bid of 1, the variance, given a value of 0, is quite high. The variance then decreases with increasing value, and for values greater than 0.5 it rises again. A possible explanation for this behaviour is that for maximal bids and fixed values, the utility fully depends on the other bidders bids. When bidding the maximum value possible and having a valuation 0, it makes sense that the variance is high since if no-one bids more, then the bidder wins the item for a high price but does not get any value out of it and ends up with a huge negative utility. However if another bidder bids more, then the bidder has luck and ends up with a utility of 0. When the bidder has a value of 1 and bids 1 then it is possible that he wins but does not have to pay quite as much (depending on the mechanism) and therefore can create positive utility. Since the value is as high as possible, this utility can be high as well. If he loses, however, the utility is 0. Therefore, the variance is quite high. For more mid-range valuations the positive and the negative utilities possible are rather small and therefore the variance between winning the item and losing it is small as well.

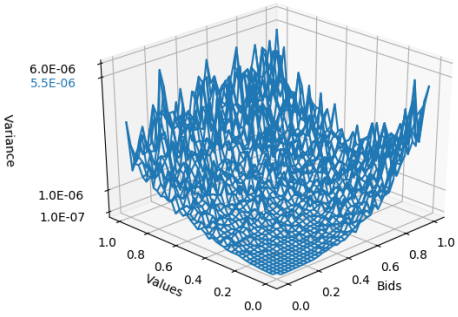
[9] $utility = valuation - price$



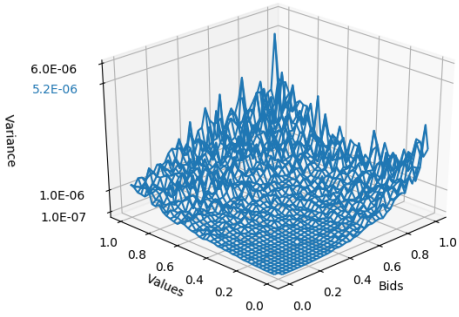
$\alpha = 1$, $\gamma = 0$ (scale: 1x)



$\alpha = 1$, $\gamma = 0.5$ (scale: 1x)

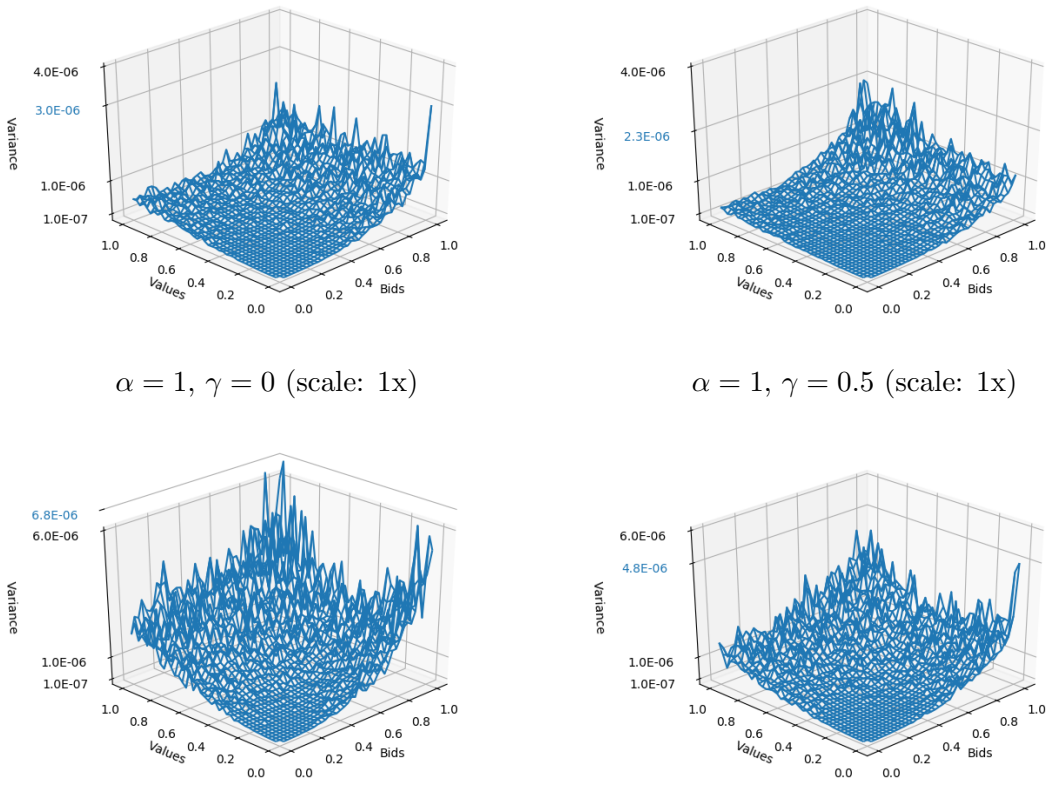


$\alpha = 2$, $\gamma = 0$ (scale: 3x)



$\alpha = 2$, $\gamma = 0.5$ (scale: 3x)

Figure 3.5: **Variance of the Nearest-Bid mechanism, in iteration 0.** This mesh-grid-plot represents the variance for the different combinations of bid and value. The default scale (z-axis) is from 0 up to $2 \cdot 10^{-6}$. For showing the full data and still being able to distinct the general shape of the curve, some of the plots are scaled up to $4 \cdot 10^{-6}$ (2x) or $6 \cdot 10^{-6}$ (3x).



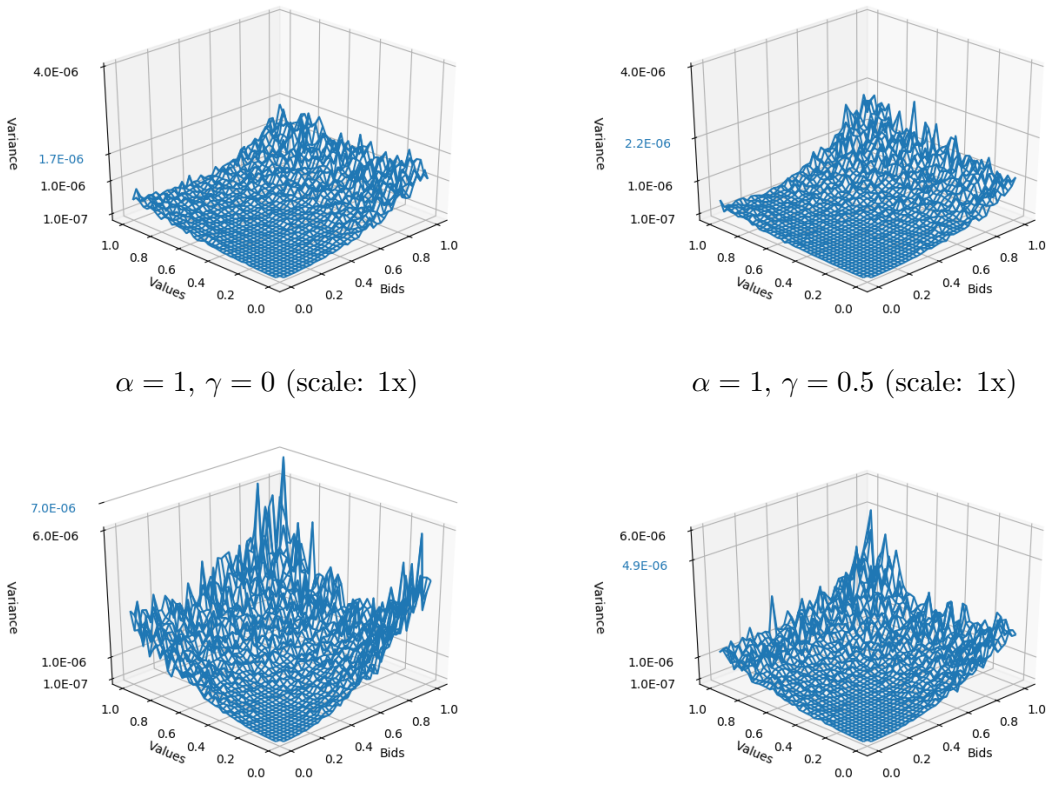
$\alpha = 1$, $\gamma = 0$ (scale: 1x)

$\alpha = 1$, $\gamma = 0.5$ (scale: 1x)

$\alpha = 2$, $\gamma = 0$ (scale: 3x)

$\alpha = 2$, $\gamma = 0.5$ (scale: 3x)

Figure 3.6: **Variance of the Proportional mechanism, in iteration 0.** This mesh-grid-plot represents the variance for the different combinations of bid and value. The default scale (z-axis) is from 0 up to $2 \cdot 10^{-6}$. For showing the full data and still being able to distinct the general shape of the curve, some of the plots are scaled up to $4 \cdot 10^{-6}$ (2x) or $6 \cdot 10^{-6}$ (3x).



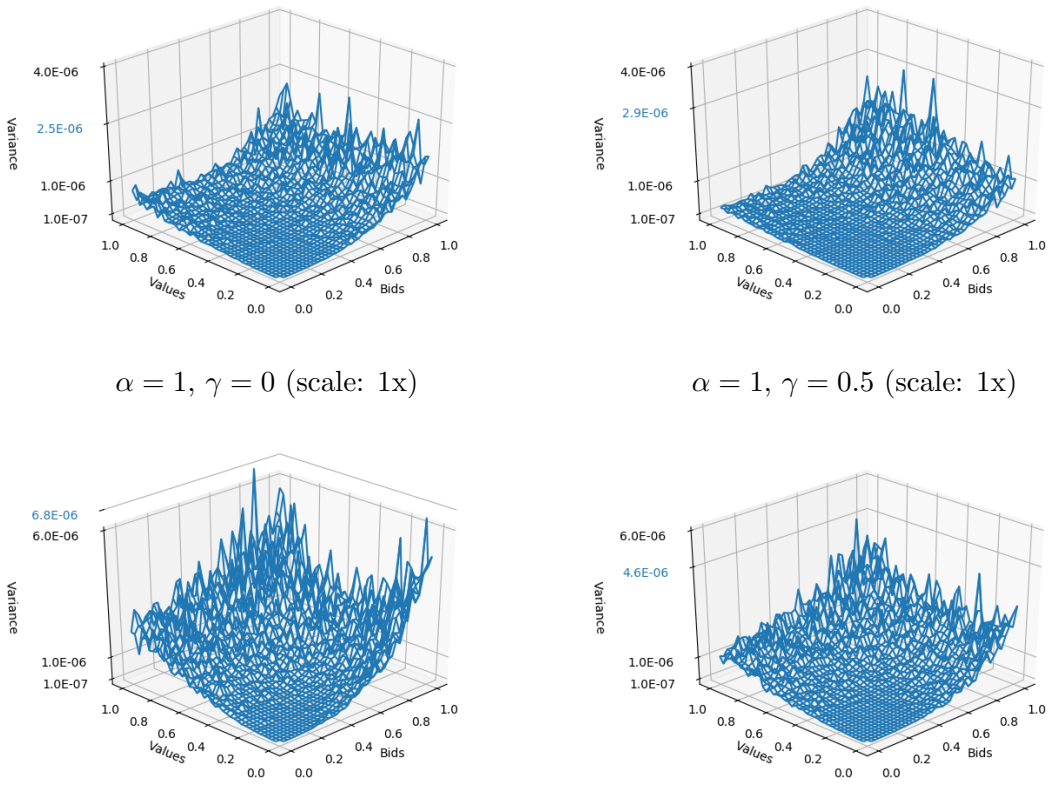
$\alpha = 1$, $\gamma = 0$ (scale: 1x)

$\alpha = 1$, $\gamma = 0.5$ (scale: 1x)

$\alpha = 2$, $\gamma = 0$ (scale: 3x)

$\alpha = 2$, $\gamma = 0.5$ (scale: 3x)

Figure 3.7: **Variance of the Proxy mechanism, in iteration 0.** This mesh-grid-plot represents the variance for the different combinations of bid and value. The default scale (z-axis) is from 0 up to $2 \cdot 10^{-6}$. For showing the full data and still being able to distinct the general shape of the curve, some of the plots are scaled up to $4 \cdot 10^{-6}$ (2x) or $6 \cdot 10^{-6}$ (3x).



$\alpha = 1$, $\gamma = 0$ (scale: 1x)

$\alpha = 1$, $\gamma = 0.5$ (scale: 1x)

$\alpha = 2$, $\gamma = 0$ (scale: 3x)

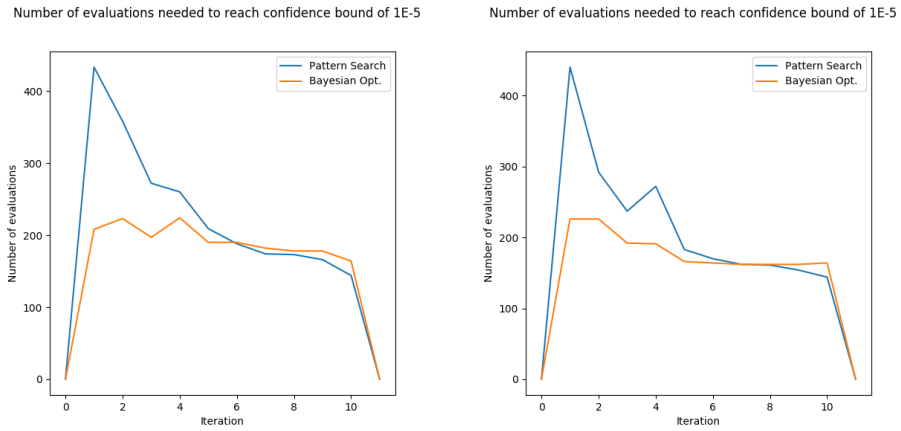
$\alpha = 2$, $\gamma = 0.5$ (scale: 3x)

Figure 3.8: **Variance of the Quadratic mechanism, in iteration 0.** This mesh-grid-plot represents the variance for the different combinations of bid and value. The default scale (z-axis) is from 0 up to $2 \cdot 10^{-6}$. For showing the full data and still being able to distinct the general shape of the curve, some of the plots are scaled up to $4 \cdot 10^{-6}$ (2x) or $6 \cdot 10^{-6}$ (3x).

3.2.4 Conclusions

The general shape of the variance for the 16 domains in focus does not differ considerably. Concluding, the same model could be applied to all of them with only small adjustments of some parameters. However, in a practical environment where we do not have any prior information and expectations about the shape of the variance for new domains, an additional effort has to be made to gather the needed information about the variance. Also, since the variance seems to be quite low for the setting of 10'000 samples, we can conclude that a rough educated estimate of the variance does suffice. During development of the source code some empirical testing has been done, which leads to the conclusion that with a fixed value for the variance, the convergence speed does not differ a lot. To verify, we pick the Proxy domain with $\alpha = 2$ and $\gamma = 0$ and run it with a variance of 10^{-7} which is lower than most of the calculated variances. The convergence of the BO with that fixed and very low variance is of similar shape[10] (see Figure 3.9). So for the current configuration, the variance does not have a big impact on the performance. However, it is reasonable to include the variance in the Gaussian process, due to the nature of the MC integration.
A greater potential for improving the performance is adjusting the sample size. A reduction of the sample size also reduces the runtime of the MC integration and therefore, the best response calculation directly. In this case, the variance most likely will grow and thus its importance increases. When the variance is brought into relation with the sample size, a smaller sized sample batch could be compensated by adjusting the variance and an optimal sample size can be configured in a way to optimize the relationship between runtime and accuracy.

[10] The small differences in the shape of the graphs can be explained by the fact that the two plots originate from different experiment runs and therefore they worked with different sample sets. Nevertheless, the visible similarity in plots shows that the change in variance, did not change the outcome drastically.



calculated Variance fix Variance ($1E-7$)

Figure 3.9: **Needed evaluations per iteration for Proxy $\alpha = 2, \gamma = 0$, BO is strategy-defining, calculated and fixed variance.** The graphs on the left how how many evaluations are needed in total over all 40 point, per iteration, when the variance is exactly calculated. To derive the noise we take the difference of the expected utility resulting from the MC integrations and the ground truth result. The plot on the right shows the same but for a fixed variance of 10^{-7}.

3.3 Bayesian Optimization Analysis

The main idea behind this experiment is to validate the results produced by BO. Until now, we only focused on the performance, hence the convergence and runtime effects of the optimization, and not on the quality. Therefore we will explore and compare the final strategies produced by the optimizers with the respective analytical results derived by [Ausubel and Baranov, 2018].
BO selects the next point of evaluation by itself according to the acquisition function. This process runs in the background and in the normal use-case we do not see what points get evaluated. Also, in the normal CA-BNE algorithm, we do not see the individual optima found by the best response algorithm but only the strategy which is put together from all the optima. To verify that the BO does indeed work as expected and the result is trustworthy, we compare it to the best responses found by the PS and the result calculated by a grid search.

3.3.1 Experimental Setup

In the course of this experiment, we compare the final strategies. We only extract the final strategy, which was written in the output file, and compare the results with the analytical result derived by [Ausubel and Baranov, 2018]. However in [Ausubel and Bara-

nov, 2018] the analytical results are only written down for $\alpha = 1$. To be able to get the analytical results for all 16 domains, the formulations from the works of [Bosshard et al., 2017] are used which are contained in the code of the CA-BNE ([Bosshard et al., 2018]). For better comprehensibility, the pseudo-code of the analytical-result-calculations is included in the appendix A.3.1.

In the other part of this experiment, we take a deeper look into the behind-the-scenes of the BO. To be able to analyse the BO, we add a callback to the optimization of the acquisition function and write the whole acquisition function into a file so that we can plot it as a graph[11]. For the calculations the results of the Gaussian process are needed anyways, therefore we can note the expected utility and the variance of that expected utility in the same file. Another callback we apply to the `BayesianOptimization` module, where all the trainings points that the BO needed, are saved. Further, we calculate the optimum of the utility function with the PS and a grid search. Note that we write a individual file for these intermediate results for all different valuations. Note that BO has multiple steps to it, so we could even make an animation where we see which next evaluation point got added in one iteration. However, we are mainly interested in the final result. Therefore, we just plot the state at the end of a BO execution.
In addition to the callback results, the final strategies at the end of one BNE-iteration are also saved in a file.

3.3.2 Configuration

For the grid search, the result of which acts as a benchmark for the optimum of the acquisition function, we use a grid with 1000 points. The configuration for the PS stays as it is in [Bosshard et al., 2017]. For the BO we set the maximum number of iterations to 25. For everything else we keep the configurations to the default one. So, 10'000 MC samples are used and the length-scale parameters are set to 1.0 for the local and 0.6 for the global bidders. In contrast to prior experiments we add more callbacks to the context as discussed before.

3.3.3 Results

First, we will discuss the results of the comparison of the final strategies produced by PS and BO and the respective analytical solution. After that, we look at the plots that illustrate the functionality of the BO and discuss any anomalies that appear.

Quality of final strategies

We will discuss the representation of the results in the example of the Quadratic mechanism with $\alpha = 1$ and $\gamma = 0.5$ (see Figure 3.10). On the x-axis the valuations of the

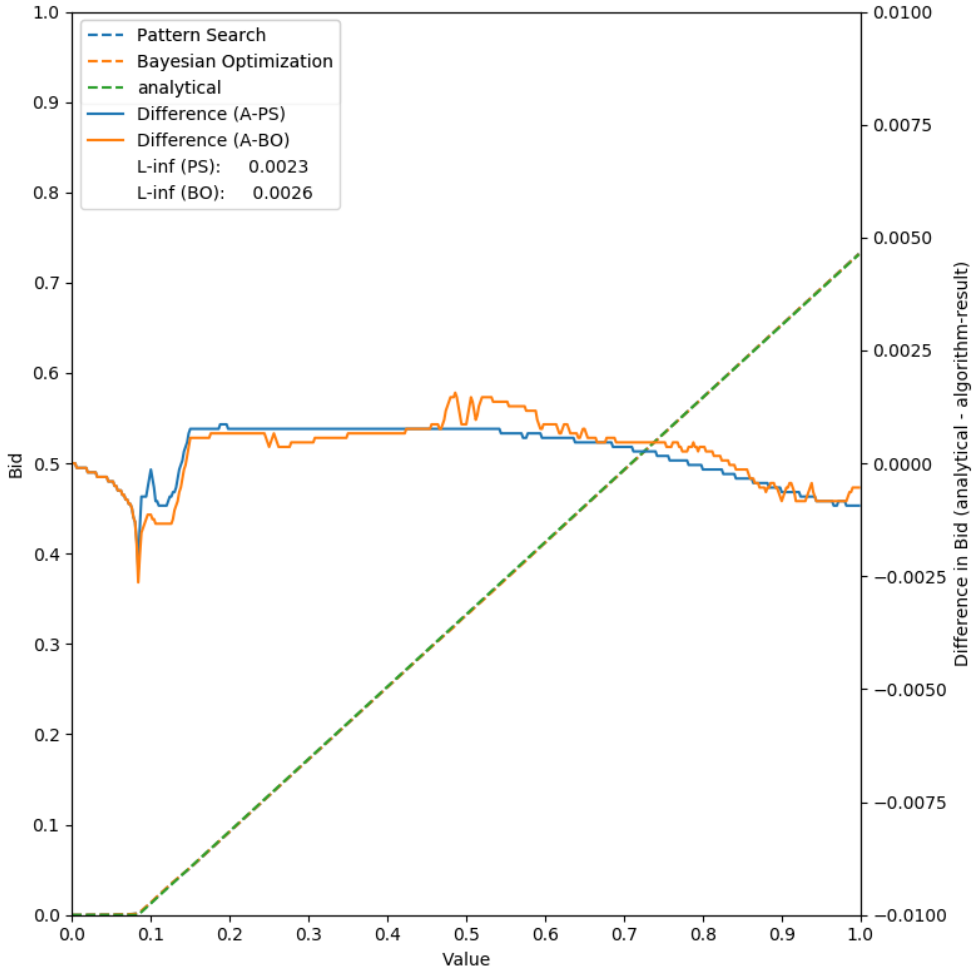
[11] to be able to see the acquisition function in the combined plot we scale it by 1000

bidder are displayed. The dotted lines in blue, orange and green display the shape of the strategy produced by the PS, the BO and the analytical results, with respect to the y-axis.
Thought out of all of the 16 domains, the differences between BO, PS and the analytical results are barely noticeable on this scale. To counter that, we calculate the difference between the analytical results and the results from the BO and PS and plot them as blue and orange solid lines. The respective scale is visible on the right y-axis. When plotting the difference, we show the effective difference and not only the absolute, which makes negative values visible as well. In addition to having only a plot that shows the shape, we also calculate the L_∞-distance[12] between the empirical results (PS and BO) and the analytical result. These results are displayed in the legend in the top left of the plot.

Looking at all the plots (Figure A.33 - A.36), we see that for the Quadratic (Figure A.36) and the Proportional (Figure A.34) mechanisms, the strategies produced by the BO are reasonably close to the ones produced by the PS. Additionally, PS and BO deviate from the analytical result in a similar shape and form. The BO, however, is rather inconsistent and oscillates around the analytical result. It makes sense that the Quadratic and the Proportional mechanism do not differ in their quality of result, since these two mechanisms work similarly.
For the Nearest-Bid (Figure 3.11) and Proxy (Figure 3.12) mechanisms the BO does perform considerably worse than the PS. Especially in the Proxy mechanism, where not only the L_∞ is quite high but also the difference between the analytical result and empirical result oscillates a lot and is very inconsistent.

Leaving the main-domains and entering the level of the subdomains, we see that for the well-performing domains (Proportional and Quadratic) no considerable difference between the sub-domains can be noticed. For the worse performing domains (Proxy and Nearest-Bid) however, in the subdomains with correlation ($\gamma = 0.5$), BO produces results deviating from the analytical result much more than the PS. We see that for $\gamma = 0$, the shape of the orange solid line is more or less the same as the blue one. For $\gamma = 0.5$, there are some really large peaks and the general shape differs considerably. In Table 3.3 we noted all the L_∞-distances for all the 16 domains, for PS and for BO. We see that our results for PS are similar to the ones noted in [Bosshard et al., 2017]. We also notice that for the Proportional and Quadratic mechanism the L_∞-distance is pretty much the same for all sub-domains except for $\alpha = 2, \gamma = 0$.

[12] Also called Chebyshev distance



$\alpha = 1, \gamma = 0.5$

Figure 3.10: **Comparison of final strategy with the analytical result (Quadratic).** The dotted lines represent the strategies for BO, PS and the analytical result. The bold lines show the difference of the result of the BO and PS compared to the analytical solution. The difference is defined by analytical result minus result from BP or PS. The domain in use here, is the Quadratic mechanism with $\alpha = 1, \gamma = 0.5$

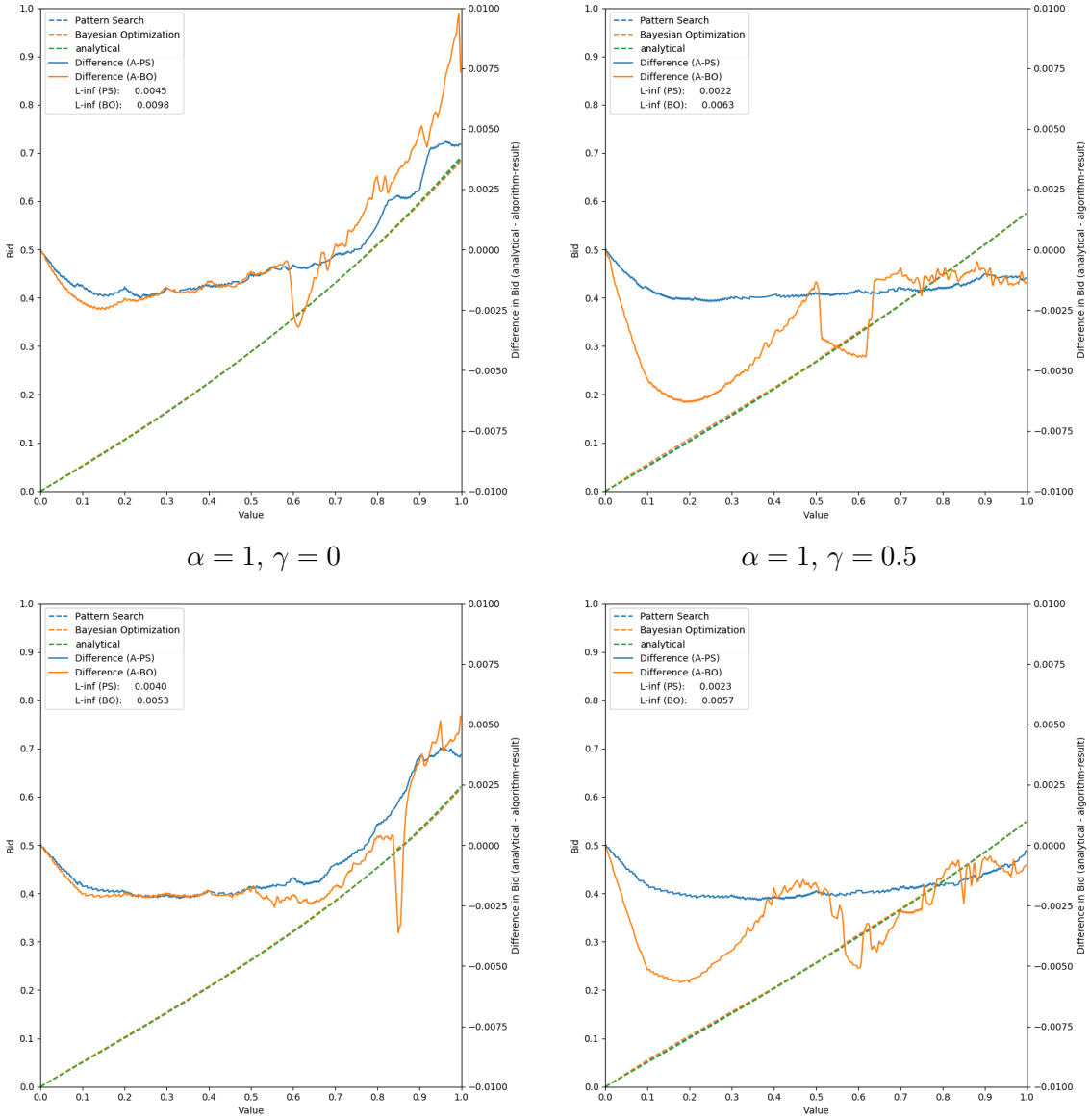
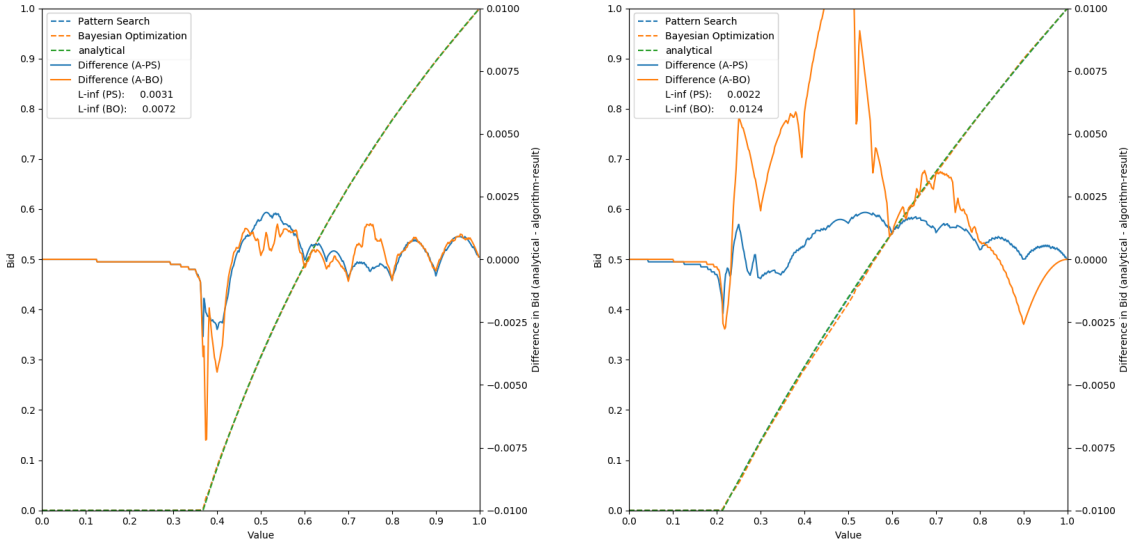
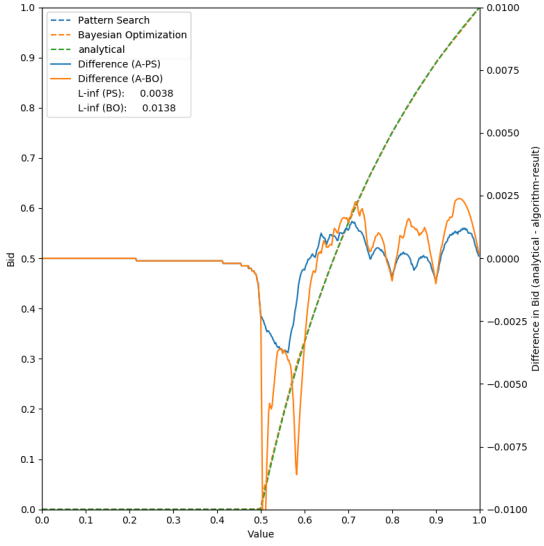


Figure 3.11: **Comparison of final strategy with the analytical result (Nearest-Bid).** The dotted lines represent the strategies for BO, PS and the analytical result. The bold lines show the difference of the result of the BO and PS compared to the analytical solution. The difference is defined by analytical result minus result from BP or PS. The domain in use here is the Nearest-Bid mechanism and all four sub-domains are displayed.

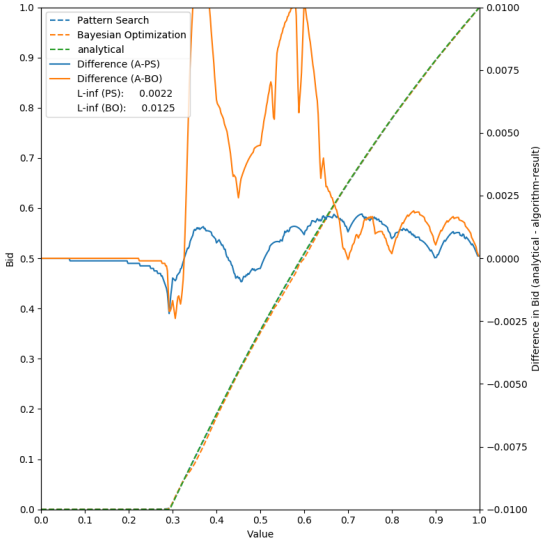


$\alpha = 1, \gamma = 0$

$\alpha = 1, \gamma = 0.5$



$\alpha = 2, \gamma = 0$



$\alpha = 2, \gamma = 0.5$

Figure 3.12: **Comparison of final strategy with the analytical result (Proxy).** The dotted lines represent the strategies for BO, PS and the analytical result. The bold lines show the difference of the result of the BO and PS compared to the analytical solution. The difference is defined by analytical result minus result from BP or PS. The domain in use here is the Proxy mechanism and all four sub-domains are displayed.

mechanism	optimizer	α=1/γ=0	α=1/γ=0.5	α=2/γ=0	α=2/γ=0.5
NearestBid	PS	0.0045	0.0022	0.0040	0.0023
	BO	0.0098	0.0063	0.0053	0.0057
Proportional	PS	0.0030	0.0023	0.0019	0.0014
	BO	0.0045	0.0023	0.0061	0.0015
Proxy	PS	0.0031	0.0022	0.0038	0.0022
	BO	0.0072	0.0124	0.0138	0.0125
Quadratic	PS	0.0030	0.0023	0.0019	0.0014
	BO	0.0033	0.0026	0.0061	0.0015

Table 3.3: **L_∞-distances.** In this table we display the L_∞-distances for all the subdomains of the 4 mechanisms. The distances are calculated for BO and for PS.

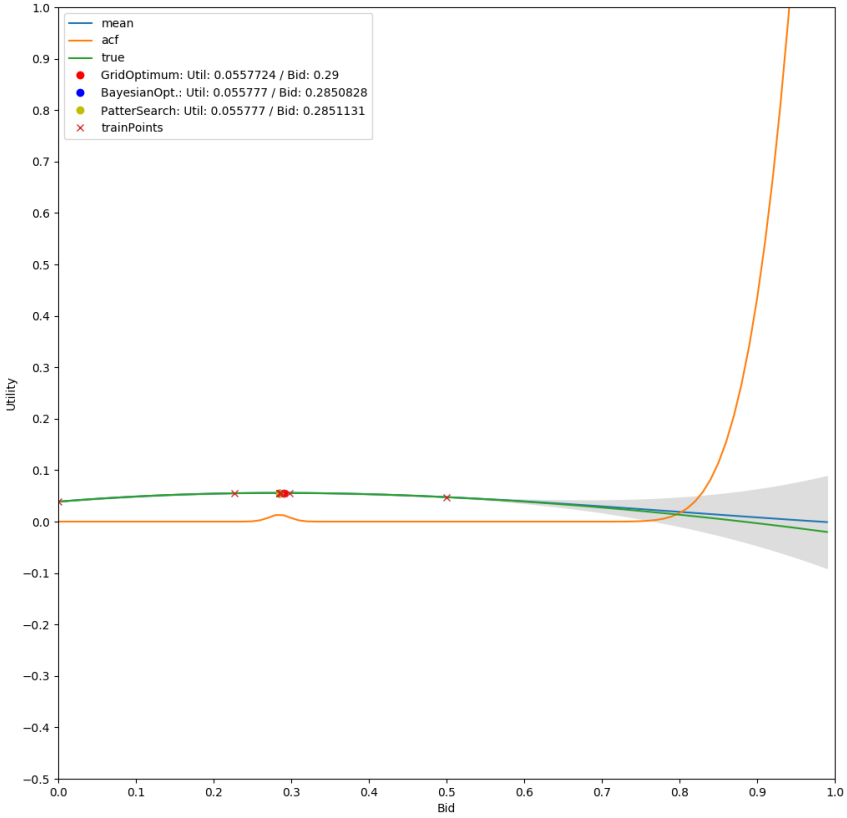
BO Illustration

Looking through the generated plots[13] where the steps of the BO are visible, there is something that draws attention. For some of the plots, the acquisition function is quite large on the right edge of the plot. We will discuss this case on the example of Figure 3.13. We see that the expected utility (mean) in blue fits the true function in green quite well for low bids. With rising bids, we see that these two graphs start to deviate and also the variance represented by the grey area ([mean $\pm$ 2 standard-deviations]) grows. Looking at the acquisition function we see there is a bump where the evaluations (red crosses) are made and to the right the acquisition function explodes. Remember the acquisition function is scaled by factor 1000. Looking at the legend, we see that the found optimum for grid search, BO and PS are similar. However, looking at where the BO made the evaluations during the search, there is something that is suspicious. We see that to the right, where the acquisition function explodes and the variance-area spreads, there was no evaluation made. As we saw in chapter 2 the BO should evaluate at the maximum of the acquisition function, which in this case is obviously to the right.[14]
So in Figure 3.13, we see that the optimization of the acquisition function clearly does not work like intended and somehow gets stuck at a local maximum. In this particular case this is not a big issue since the correct optimum is still found, nevertheless, it is important to find the source of this problem.
We are using Apaches BrentSearch [The Apache Software Foundation, 2016a] for the optimization of the acquisition function since it is rather fast. As a downside, it can get stuck at local optima when the simple configuration with the least number of parameter is used and that is what happened here.

What happens when we use a different acquisition-function-optimizer, for example a

[13] These plots are only included on the CD which is handed in with this thesis because of the enormous number of plot and the little importance of most of them.

[14] Ignore the fact that this plot is from the last iteration of the BO. The first six iterations of the whole sequence can found in the Appendix on Figure A.37 and this problem appears much earlier.



Nearest-Bid $\alpha = 1$, $\gamma = 0.5$ for value 0.5 in iteration 2

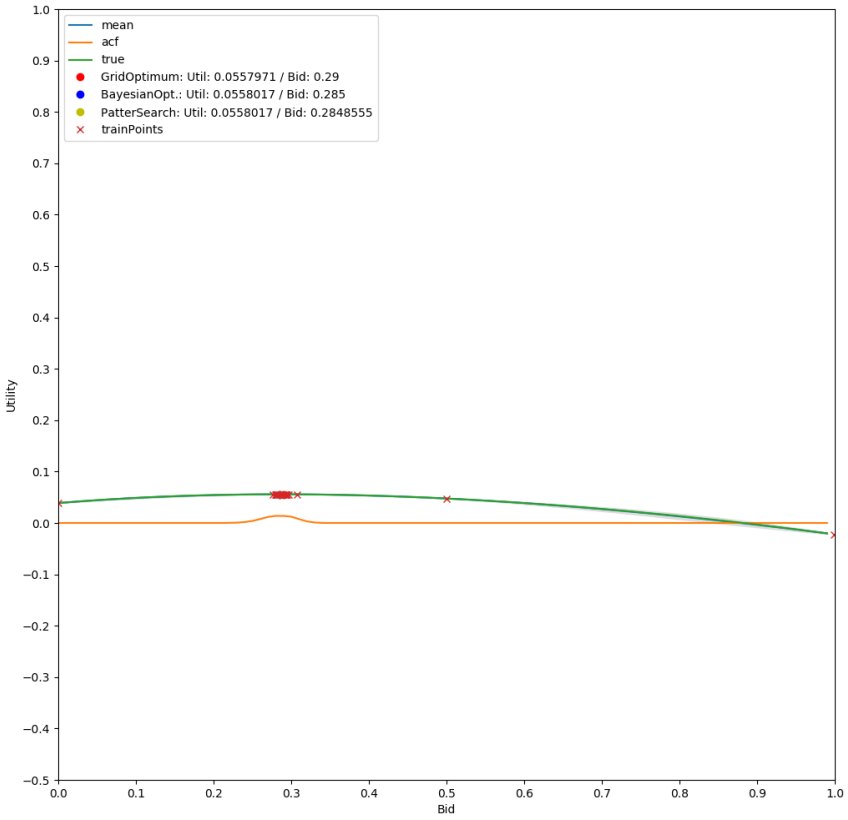
Figure 3.13: **Intermediate result of BO with Brent search.** This plot shows the intermediate result of the BO process. The blue line displays the estimated mean for the expected utility, the green graph shows the true function. The orange graph represents the acquisition function scaled by 1000. Additionally, the optima that are found by BO and PS are shown, as well as a benchmark result for which grid search was used. The red crosses indicate where the BO evaluated the expected utility function. The grey are marks the region [mean $\pm$ 2 standard-deviations].

grid search? We run the same experiment again and this time instead of using the `ACQBrentOptimizer` we use the simple `UnivariateGridOptimizer` for the acquisition-function-optimizing. Now for the same value (0.5), the plot looks more like we would expect, see Figure 3.14. We see there is an evaluation for bid 1.0 and there is only a bump in the acquisition function and the variance area does not spread anymore.
Now that we recognized this problem of using the `ACQBrentOptimizer` we should also inspect whether this change has an effect on the generated strategies. We remember that for Proxy and Nearest-Bid the final strategies generated by BO differed quite a lot from the analytical result. We will check now whether switching from `ACQBrentOptimizer` to `UnivariateGridOptimizer` has any effect on that behaviour.

For the Proxy mechanism (Figure 3.16), it does not have that big of an impact. The plots look very similar and also the L_∞-distance is not smaller than the version of the algorithm where the `ACQBrentOptimizer` is used.
For the Nearest-Bid rules (Figure 3.16), however, the final strategy for the BO does change considerably, it takes on the shape of the PS. For the sub-domains with $\gamma = 0$ the improvement is not as remarable as for $\gamma = 0.5$, where the L_∞ decreases from $0.63(\alpha = 1)$ and $0.57(\alpha = 2)$ to $0.33(\alpha = 1)$ and $0.49(\alpha = 2)$.

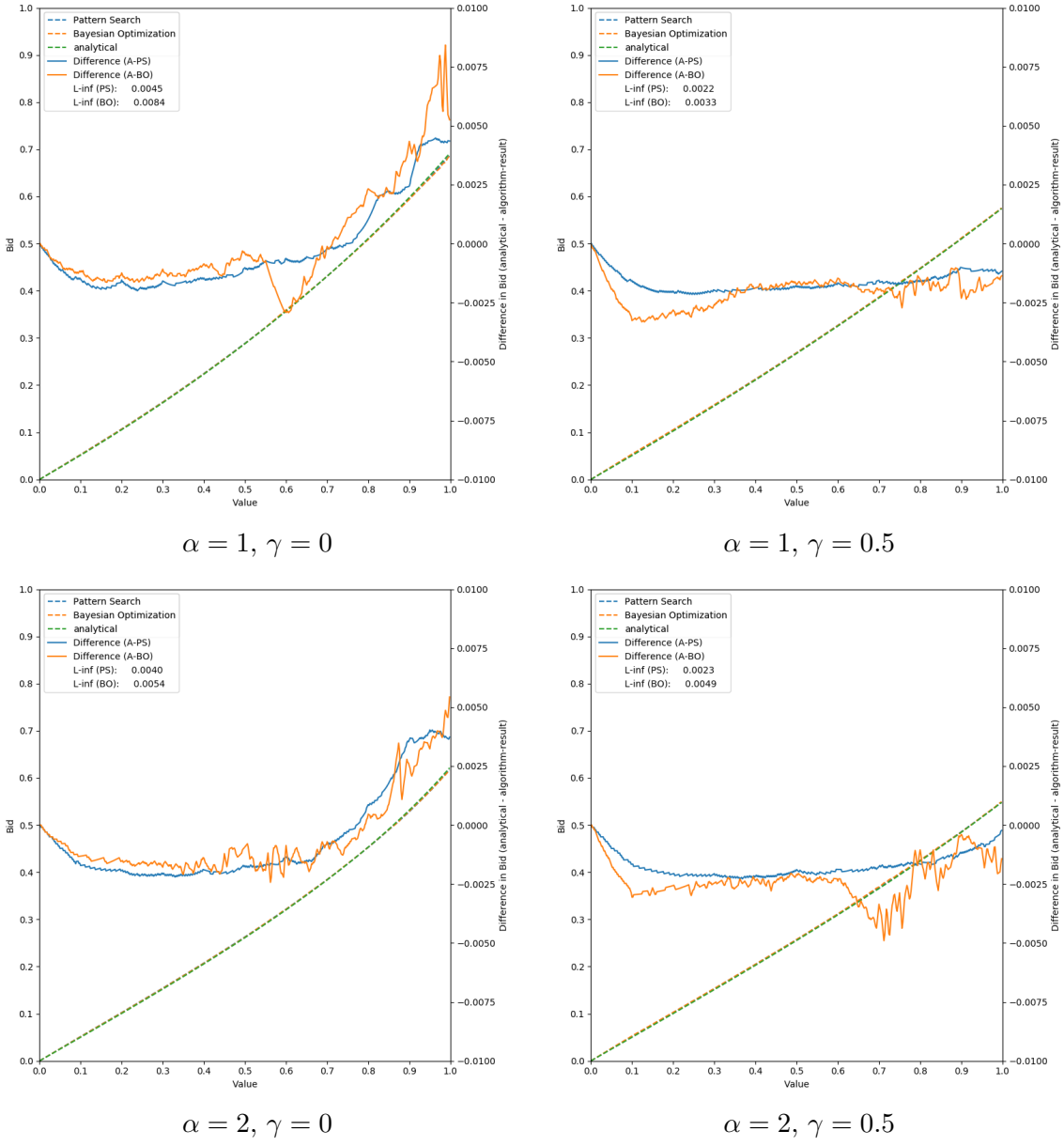
3.3.4 Conclusions

From this extended experiment we learn that it is important to keep a closer look at the behind-the-scenes of the BO. Since it is a multi-layered algorithm with many sub-routines whose intermediary results are often not directly visible, it makes sense to verify them as well. We also saw that the Brent search has its flaws when applied to the optimization of the acquisition function.
When comparing the final strategies produced by the BO to the analytical solutions, the results are reasonably close together and we can say that the BO does find trustworthy results. And it does so for most domains even with a non-optimal acquisition-function-optimizer. When comparing BO to PS however, it has to be stated that PS produces results which are closer to the analytical solution and which are more uniform.



Nearest-Bid $\alpha = 1$, $\gamma = 0.5$ for value 0.5 in iteration 2, grid search

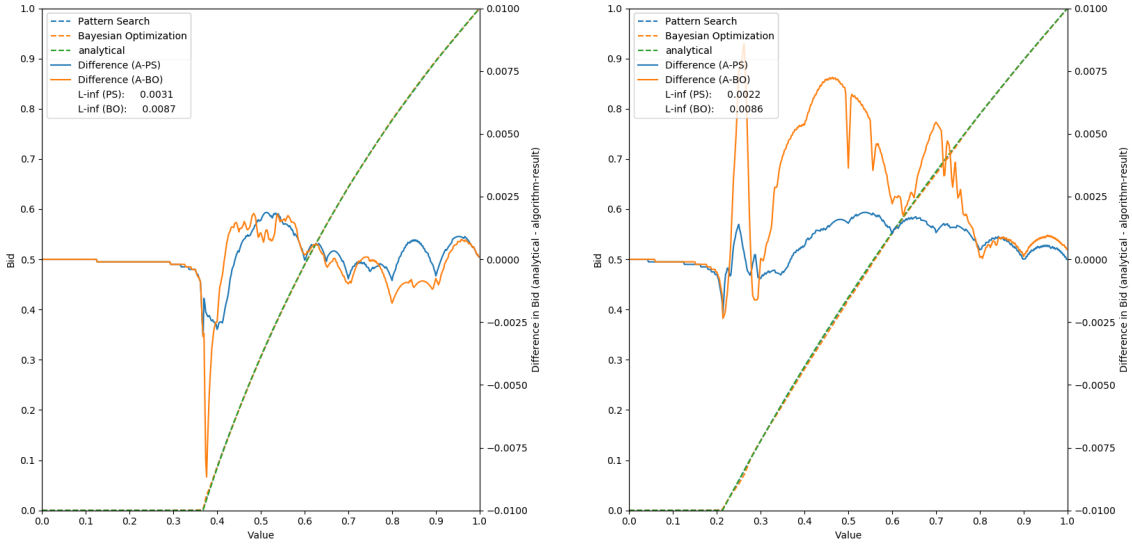
Figure 3.14: **Intermediate result of BO with Brent search.** This plot shows the intermediate result of the BO process. The blue line displays the estimated mean for the expected utility, the green graph shows the true function. The orange graph represents the acquisition function scaled by 1000. Additionally, the optima that are found by BO and PS are shown, as well as a benchmark result for which grid search was used. The red crosses indicate where the BO evaluated the expected utility function. The grey are marks the region [mean $\pm$ 2 standard-deviations]. In comparison the Figure 3.13 this time grid search was used to find the optimum of the acquisition function.



$\alpha = 1,\ \gamma = 0$ $\alpha = 1,\ \gamma = 0.5$

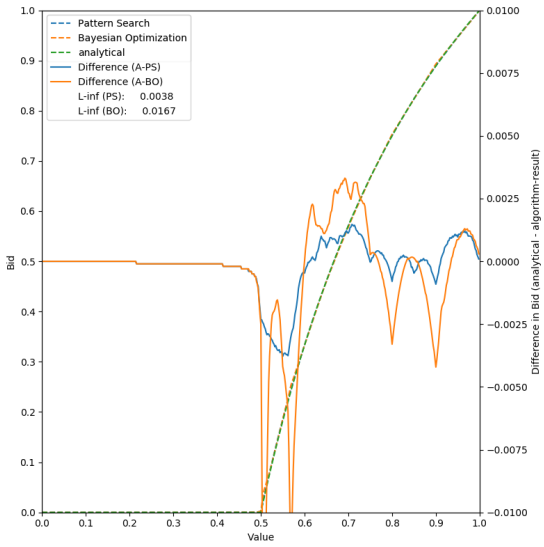
$\alpha = 2,\ \gamma = 0$ $\alpha = 2,\ \gamma = 0.5$

Figure 3.15: **Comparison of final strategy with the analytical result (Nearest-Bid), with grid search.** The dotted lines represent the strategies for BO, PS and the analytical result. The bold lines show the difference of the result of the BO and PS compared to the analytical solution. The difference is defined by analytical result minus result from BP or PS. The domain in use here is the Nearest-Bid mechanism and all four sub-domains are displayed. Compared to Figure A.33, this time we use the gird search instead of Brent search to find the maximum of the acquisition function.

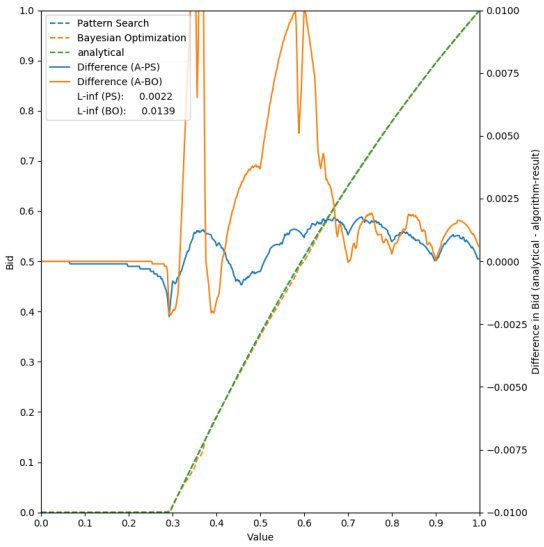


$\alpha = 1,\ \gamma = 0$

$\alpha = 1,\ \gamma = 0.5$



$\alpha = 2,\ \gamma = 0$



$\alpha = 2,\ \gamma = 0.5$

Figure 3.16: **Comparison of final strategy with the analytical result (Proxy), with grid search.** The dotted lines represent the strategies for BO, PS and the analytical result. The bold lines show the difference of the result of the BO and PS compared to the analytical solution. The difference is defined by analytical result minus result from BP or PS. The domain in use here is the Proxy mechanism and all four sub-domains are displayed. Compared to Figure A.35, this time we use the gird search instead of Brent search to find the maximum of the acquisition function.

4

Algorithm Description

In this chapter, the Java implementation of the BO algorithm is described. The implementation is written in close combination with the CA-BNE algorithm from [Bosshard et al., 2018]. Also, to stay consistent with the code from [Bosshard et al., 2018] it respects the guideline of keeping the number of dependencies as low as possible. Therefore the only dependency besides the CA-BNE is the widely used Apaches Common Math library [The Apache Software Foundation, 2016d]. This library is also a dependency for the CA-BNE algorithm.

4.1 Bayesian Optimization

The `BayesianOptimization` class is a implementation of `Optimizer` interface from the CA-BNE algorithm [Bosshard et al., 2018]. This module is responsible for finding the pointwise best response for individual values of a bidder. The main functionality of the BO is contained in the `findBR` method and is outlined in the pseudo-code below.

Algorithm 1: Bayesian Optimization

Data: bidder i, value v, Bid *currentBid*, strategies $S = s_0, ..., s_n$, Bids $B = b_0, ..., b_t$ (trainings-input)

Result: *currentBest*, utility of *currentBest*, utility of *originalBest*

```
1  maxIter := maximal number of Iterations;
2  nsamples := number of samples for the MC-Integration;
3  trainingsPoints := List of all TrainingsPoints stored as Objects with attributes
     bid, utility, variance and nsamples;
4  foreach b_i ∈ B do
5  |   (util, var) = computeExpectedUtility(i, v, b_i, S, nsamples);
6  |   add (b_i, util, var, nsamples) to trainingsPoints;
7  end
8  (util, var) = computeExpectedUtility(i, v, currentBid, S, nsamples);
9  originalBest := (currentBid, util, var, nsamples);
10 add originalBest to trainingsPoints;
11 currentBest := originalBest;
12 for ( iter = 0;  iter < maxIter;  iter+ = 1 )
13 |   acqMaximum = acqOptimum(i, S, trainingsPoints);
14 |   (util, var) = computeExpectedUtility(i, v, acqMaximum, S, nsamples);
15 |   add (acqMaximum, util, var, nsamples) to trainingsPoints;
16 |   currentBest = trainings-point with highest utility;
17 |   if breakCondition == true then
18 |   |   break;
19 |   end
20 end
```

4.1.1 Context

With the use of the `BayesianOptimizationContext` the `BayesianOptimization` can be configured in a strategy[1]-like fashion. Therefore, the following components can be configured before runtime, using the `BayesianOptimizationContext`.

`kernel`
: defines which kernel should be used in the Gaussian process. (see subsection 2.3.2)

`ACQfunction`
: defines which implementation of acquisition function to use. (see subsection 2.3.3)

`builder`
: instance of `Stringbuilder`[2] which is used for writing program output.

`randomBidSampler`
: instance of `BidSampler` (see 4.5.3) which is used for the generation of the initial trainings-points.

`lengthParameter`
: array of parameters, one for each dimension, which are used for the Gaussian process as length-scale parameter. (see subsection 2.3.1)

`acqLogger`
: Event-listener that is used to write intermediate result to output files.

`acqOptimizer`
: defines which optimizers should be used to optimize the acquisition function.

`varianceCalc`
: defines how the variance should be calculated or estimated.

`BOintegrator`
: instance of a slightly modified[3] version of the `MCIntegrator` which is part of the context so that it is a shared object and its caches can be accessed from different modules.

`update`
: boolean which can be set to `true` so that reappearing evaluation points are updated rather than added again to the array of training points..

`numberOfMCSamples`
: defines how many samples the MC integration should use.

[1] Design pattern described in [Gamma et al., 1995]
[2] defined in[Bosshard et al., 2018]
[3] The modification enables to define the number of samples the MC integrations should use at runtime.

4.1.2 Callbacks

The BO algorithm has four different anchor points for callbacks. These are assembled in the `BayesianOptimizationCallback` class. The anchor-point `startOfBO` is executed only once before line 4 in Algorithm 1. This callback can be used to run a preliminary calculation, for example for getting a ground truth. Symmetrically, `endOfBO` is run once after line 20. The other two callbacks are `beforeIteration` (after line 12, before 13) and `afterIteration` (after line 16, before 17), these callbacks are very useful for writing some intermediate results to files and for analysing the progress between the iterations of the BO.

4.1.3 Other Features

The `BayesianOptimizationContext` class can be equipped with some other features and abilities which are mostly used when analysing and verifying the results. By enabling/disabling these features the BO can be configured further.

Break Condition

By default, the BO does not have any break condition and is fully controlled by the `maxIter` parameter. For testing, experimenting and runtime improvements a custom `BreakCondition` (see 4.5.2) can be applied to the BO.

Updating of Trainings Points

The implementation of the BO allows an already evaluated point to maximize the acquisition function again and therefore to be evaluated again. Because we are considering the noise of the MC integration, a re-evaluation can lead to a different result. One way of handling multiple evaluations of the same point is by adding all evaluations as new points to the list of training points. However, it might also make sense to update the evaluation such that there are fewer training points in that list. An advantage of having less list items would be that the matrix in the Gaussian process becomes smaller, which might save calculation time. There is a boolean argument `update` which specifies whether to update reappearing points or to add them again to the list of training points if they appear to maximize the acquisition function.

Operating on a Grid

Usually, the BO operates on a continuous bid-space. However, for comparing the efficiency and quality of the results found by the BO, it is useful to compare the results to a ground truth optima. Getting the ground truth on a continuous space is very costly[4].

[4]This is exactly what the BO is intended to do rather efficient.

Therefore, it makes sense to search on a grid. On a grid, the ground truth result can be calculated by a naive grid search without any runtime explosion. Considering this use-case, there is a secondary constructor added to the BO such that an additional argument (`gridPoints`) can be given. When this secondary constructor is used, the bid-space (from 0 until the maximum valuation of that bidder) is fractioned into `gridPoints` (an integer) different points. All the training points and bids used in the algorithm are then rounded to the next closest grid point.

4.2 AQCOptimizer - Acquisition Function Optimization

This is the module in charge of optimizing the acquisition function. The `ACQOptimizer` is abstract. Due to major differences between the one- and multi-dimensional setting, there are different implementations needed. The root of these distinctions is that the one-dimensional case [Bosshard et al., 2018] works with `Double` as the type for bids and values, whereas in the multidimensional case it works with arrays of Doubles, so with `Double[]`. This leads to major differences in the implementation. The method `acqOptimum` directly calls the `optimize` function of the `ACQOptimizer` which is defined by the concrete implementation of the respective optimizer. There are several optimizer implemented.

`UnivariateACQBrentOptimizer`
: uses the BrentOptimizer module of Apaches Commons [The Apache Software Foundation, 2016a].

`UnivariateACQPatternSearch`
: uses a slight variation[5] of the pattern search implemented in [Bosshard et al., 2018].

`UnivariateACQGridOptimizer`
: is a normal grid search that evaluates functions in equidistant points and searches for the maximum among all of them.

`MultivariateACQNelderMeadOptimization`
: uses the NelderMeadSimplex [The Apache Software Foundation, 2016b] for the multivariate case.

All implementations share a similar structure:

[5] The modifications made are so that it can be applied to acqusition functions.

Algorithm 2: ACQ Optimization

Data: bidder i, strategies $S = \{s_0, ..., s_n\}$, Bids $B = \{b_0, ..., b_t\}$ (trainings-data)
Result: bid of *optimum*

1 *bound* := value boundaries for the bidder i;
2 *incumbent* = `getIncumbent`($trainingsdata, lengthParam$);
3 *objFunc* := `ACQObjectiveFunction`(trainingsData, context,bound, incumbent);
4 *optimum* = optimum of *objFunc* found using search-strategy

where `getIncumbent` returns that training point, that has the largest mean in the Gaussian process. The *objFunc* assigns the respective acquisition value to all bids.

4.3 Gaussian Process

The Gaussian Process is where much of the work is done. Like explained in section 2.3.1, a Gaussian distribution is calculated over possible utility functions going through the training points. As a result we get, for each bid, a mean for the expected utility as well as a variance of the result. These variance- and mean-values are then used by the `ACQObjectiveFunction` module to assign the acquisition-value, which is a combination of mean, variance and other parameters defined by the selected acquisition function.

Algorithm 3: Gaussian Process

Data: TrainingsPoints $P = p_0, ..., p_t$ (trainings-data), Bid testData, double lengthParam
Result: bid of *optimum*

```
1  targets := RealVector containing utilities of trainingsPoints;
2  n := number of trainingspoints;
3  covMatrix := nxn-Matrix;
4  for ( i = 0;  i < n;  i+ = 1 )
5  |  for ( j = 0;  j <= i;  j+ = 1 )
6  |  |  k := kernel.solve(bid of p_i,bid of p_j,lengthParam);
7  |  |  covMatrix(i,j) = k;
8  |  |  covMatrix(j,i) = k;
9  |  end
10 |  covMatrix(i,i) += variance of p_i;
11 end
12 k_* := Vector of dimension n;
13 for ( i = 0;  i < n;  i+ = 1 )
14 |  k_*(i) = kernel.solve(bid of p_i,testData,lengthParam);
15 end
16 α := CholeskyDecomosition(covMatrix).solve(targets));
17 mean = k_*.dotProduct(α);
18 L (Matrix) := CholeskyDecomosition(covMatrix).getL());
19 v (vector) := LUDecomposition(L).solve(k_*));
20 variance = kernel.solve(testData,testData,lengthParam)-v.dotProduct(v);
```

where `kernel.solve` calculates the covariance according to the assigned kernel. `dotProduct`, `CholeskyDecomosition` and `LUDecomposition` are modules from [The Apache Software Foundation, 2016d] that do exactly what their names imply.

4.4 ACQ-function and ACQObjectiveFunction

The acquisition functions described in subsection 2.3.3 implement the interface `ACQ` which combines the mean and variance from the Gaussian process into one acquisition-value. These acquisition functions can then be wrapped into `ACQObjectiveFunction` which we need to be compatible with [The Apache Software Foundation, 2016d]. Like this we are able to use optimizers from [The Apache Software Foundation, 2016d].

4.5 Other Objects

4.5.1 TrainingsPoint Object

The BO is based on training points. Based on them, the next point to evaluate is selected. To store all relevant pieces of information about such a training point, a new object is created - the `TrainingsPoint` object.
The first two attributes of this object are bid and utility. Since we are also considering the noise of the MC integrations for our implementation, the third attribute is the variance of that niose. Like this, it is possible to separate more accurate evaluations form very basic ones. For analytical reasons, the number of MC samples is also saved in the object. Thus, the direct relation between evaluation cost and the evaluation quality is directly stored. The more samples the more accurate the result should be but also more costly to evaluate.

4.5.2 BreakCondition

The `BreakCondition` interface contains a method `checkBreakCondition` that returns a boolean which is `true` if the implemented break-condition is met and `false` otherwise. Currently there is only one implementation of this interface. The `CompareToTrue` break-condition compares the found utility against a groundbtruth value and is used for the experiment explained in section 3.1.

4.5.3 BidSampler

The `BidSampler` is used to define the first (initial) training points we use for the BO. This abstract class can have multiple implementations for the univariate, but also for the multivariate case. Configuring the `BidSampler` we could exactly define with which trainings points to start the BO process.

4.6 Original BNE-Algorithm

In this section, some of the key classes of the original BNE-algorithm by [Bosshard et al., 2017] are explained in more details. The collection is limited to classes that are affecting

the work of this thesis. For a deeper insight into the algorithm, we refer to [Bosshard et al., 2017] and [Bosshard et al., 2018].

Optimizer

The `Optimizer` defines which optimizer should be used for the best response calculation. By default, this is the PS algorithm. This is where the substitutional BO algorithm could be selected.

RandomGenerators

The `RandomGenerator` is used to generate random numbers which are used as valuations for the bidders. It is mainly used for the MC integration where many samples are drawn and for these samples all opposing bidder need a random valuation assigned to them, such that from the according bid the utility can be calculated. There are three different implementations of this generator and it is important to understand what the difference is.

`NaiveRandomGenerator`
: This implementation uses plain random numbers[6] which are not related at all.

`QuasiRandomGenerator`
: This generator makes use of the [The Apache Software Foundation, 2016c] which generates a sequences with low discrepancy which cover the sampling-region more evenly. [Bosshard et al., 2017]

`CommonRandomGenerator`
: When using this generator then the random numbers are stored in a cache and can be retrieved later on. This enables us to run the algorithm twice with the same exact random numbers.

[6] generated by the default `Math.random()`.

5

Conclusions

In the course of the work for this thesis, a module was written that extends the CA-BNE algorithm by [Bosshard et al., 2017]. This module implements the Bayesian optimization algorithm, which is then used for the best response calculation in the CA-BNE algorithm. The code written focuses on the univariate use case but implements the most important interfaces for the multivariate case as well. The code is held modular and is equipped with multiple variants for many subroutines.
Using this extension, empirical experiments were run, whose results are used to explore the possibilities of using BO for improving the performance of the BNE-CA algorithm.

Based on the number of evaluations the best response algorithms need to make until reaching an area around the true optima, the BO performs better than the PS in the domain under research. Due to its more advanced determination of the next evaluation point, the BO can explore the entire region faster, and decreases the chances of getting stuck on a local optimum. As a consequence, the starting points for the BO matter less compared to the PS. However, the PS does use the best response from the last iteration as the starting point, which does not require any additional computation and is a reasonably good starting point as well. Nevertheless, BO saves on utility function evaluations, especially in the first few BNE-iterations and therefore produces a runtime improvement, if the utility function evaluation is computationally more complicated than the BO itself.

Because BO combines multiple subroutines that can have different variants and that require some parameter input, it is highly configurable, which increases the difficulty of using the algorithm, but also enables configuring it more specifically. Finding the optimal combination can produce an even bigger improvement, but is also harder to find. The different subroutines have to work together reliably and one incorrect subroutine can endanger the quality of the final result.

Exploring the effect of the variance of the noisy observations resulting from the MC integration on the convergence of the BO did not produce any ground-shaking discoveries. However, BO is able to include the concept of noise and variance in the calculation and therefore has the potential for performance and quality improvement. Further research is needed in this matter.

From the results of this thesis, short-term performance improvements can be suggested. By reducing the best response iteration limit in later BNE-iterations, the runtime can be reduced without affecting the quality of the result noticeably. This measure can be applied to the BO as well as the PS. Using BO would also make it possible to implement dynamic stopping criteria to prevent useless evaluations.

The domains under research in this thesis do not command complex enough utility function evaluations to allow a statement that the BO clearly outperforms the PS on these domains. Nevertheless, with further research and (hyper-)parameter optimization, the BO constitutes a viable alternative to the PS.

6

Future Work

There are many opportunities to extend the work of this thesis. A lot of effort was put into the creation of the code base for the BO extension to the CA-BNE algorithm. In the field of BO, the configuration of the parameters is an important but time consuming task. There is a lot of room for improvements and research of direct relationships between the setting of the parameters and the performance and quality of the result. One direction of future work could be finding the optimal BO-configuration for the different domains, maybe going as far as applying some machine-learning approach to that.[1] For the learning of the hyperparameters in the Gaussian process, findings from [Martinez-Cantin, 2015] could be applied.

As mentioned in section 3.2, being able to determine an optimal sample size for the MC integration and incorporating the sample-size-dependent variance into BO does have the potential to decrease the runtime of the CA-BNE algorithm. Seen in 3.1, only the first few BNE-iterations need many best response iterations. Therefore, it is advised to apply a more dynamic best response iteration limit to the current CA-BNE-algorithm which decreases the limit with each additional best response iteration. Also, since the CA-BNE tries to find an equilibrium, it could be worth exploring how the results from previous iterations could be reused and discounted by increasing the variance and use them as cheap training points in the BO.

Other possible paths to improve the BO would be to develop a kernel for the Gaussian process that is adjusted to the setting of the combinatorial auction and the domains under research. In general, kernels incorporate some assumptions about the target function. For this thesis only very general assumptions were made and the Squared Exponential (Gaussian) Kernel was used. To test the modularity of the code, another frequently used kernel, the Matern kernel, was implemented. Besides the kernel, also the acquisition function can be explored to greater extent.

Conclusively, it should be mentioned that this thesis ended up focussing mainly on the univariate CA-domains. Therefore, another future work would be to explore the behaviour of the BO in the multidimensional case. The code base is already prepared

[1] Bayesian-optimization itself can also be considered as a machine-learning approach

for such research and most interfaces have a multivariate implementation, but there is still much open to be descovered. The multidimensionality causes rather long run-times and the Gaussian process used by the BO scales up with the number of dimensions as well. Therefore, possible solutions to reduce the complexity of multiple dimension could be explored, where considerable approaches would be applying the dropout method developed by [Li et al., 2017]. When extending the algorithm to multivariate functions, the optimization of the acquisition function also has to be adapted to multiple dimensions, for which the DIRECT algorithm described in [Jones et al., 1993], would be a fitting approach.

References

[Ausubel and Baranov, 2018] Ausubel, L. M. and Baranov, O. V. (February 2018). Core-selecting auctions with incomplete information. Working Paper.

[Ausubel et al., 2006] Ausubel, L. M., Milgrom, P., et al. (2006). The lovely but lonely vickrey auction. *Combinatorial auctions*, 17:22–26.

[Ausubel and Milgrom, 2002] Ausubel, L. M. and Milgrom, P. R. (2002). Ascending auctions with package bidding. *Advances in Theoretical Economics*, 1(1).

[Blumrosen and Nisan, 2007] Blumrosen, L. and Nisan, N. (2007). Combinatorial auctions. *Algorithmic game theory*, 267:300.

[Bosshard et al., 2017] Bosshard, V., Bünz, B., Lubin, B., and Seuken, S. (2017). Computing bayes-nash equilibria in combinatorial auctions with continuous value and action spaces. In *IJCAI*, pages 119–127. ijcai.org.

[Bosshard et al., 2018] Bosshard, V., Bünz, B., Lubin, B., and Seuken, S. (2018). CA-BNE repository. [Retrieved September 2, 2018, from https://github.com/algorias/CA-BNE].

[Brochu et al., 2010] Brochu, E., Cora, V. M., and De Freitas, N. (2010). A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*.

[Cramton, 2013] Cramton, P. (2013). Spectrum auction design. *Review of Industrial Organization*, 42(2):161–190.

[Day and Cramton, 2012] Day, R. W. and Cramton, P. (2012). Quadratic core-selecting payment rules for combinatorial auctions. *Operations Research*, 60(3):588–603.

[Day and Raghavan, 2007] Day, R. W. and Raghavan, S. (2007). Fair payments for efficient allocations in public sector combinatorial auctions. *Management Science*, 53(9):1389–1406.

[Gamma et al., 1995] Gamma, E., Helm, R., Johnson, R. E., and Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional Computing Series. Addison-Wesley, Reading, MA.

[Harsanyi, 1968] Harsanyi, J. C. (1968). Games with incomplete information played by bayesian players part ii. bayesian equilibrium points. *Management Science*, 14(5):320–334.

[Jones et al., 1993] Jones, D. R., Perttunen, C. D., and Stuckman, B. E. (1993). Lipschitzian optimization without the lipschitz constant. *Journal of Optimization Theory and Applications*, 79(1):157–181.

[Jones et al., 1998] Jones, D. R., Schonlau, M., and Welch, W. J. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492.

[Kushner, 1964] Kushner, H. J. (1964). A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering*, 86(1):97–106.

[Li et al., 2017] Li, C., Gupta, S., Rana, S., Nguyen, V., Venkatesh, S., and Shilton, A. (2017). High dimensional bayesian optimization using dropout. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 2096–2102.

[Martinez-Cantin, 2015] Martinez-Cantin, R. (2015). Local nonstationarity for efficient bayesian optimization. *CoRR*, abs/1506.02080.

[Matérn, 2013] Matérn, B. (2013). *Spatial variation*, volume 36. Springer Science & Business Media.

[Mockus, 1975] Mockus, J. (1975). On bayesian methods for seeking the extremum. In *Optimization Techniques IFIP Technical Conference*, pages 400–404. Springer.

[Nash, 1951] Nash, J. (1951). Non-cooperative games. *Annals of mathematics*, pages 286–295.

[Parkes et al., 2001] Parkes, D. C., Kalagnanam, J., and Eso, M. (2001). Achieving budget-balance with vickrey-based payment schemes in exchanges. In *IJCAI*, pages 1161–1168. Morgan Kaufmann.

[Rasmussen and Williams, 2006] Rasmussen, C. E. and Williams, C. K. (2006). *Gaussian process for machine learning*. MIT press.

[Reeves and Wellman, 2012] Reeves, D. M. and Wellman, M. P. (2012). Computing best-response strategies in infinite games of incomplete information. *CoRR*, abs/1207.4171.

[Stein, 2012] Stein, M. L. (2012). *Interpolation of spatial data: some theory for kriging*. Springer Science & Business Media.

[The Apache Software Foundation, 2016a] The Apache Software Foundation (2016a). Class BrentOptimizer. [Retrieved August 30, 2018,

from http://commons.apache.org/proper/commons-math/javadocs/api-3.6/org/apache/commons/math3/optim/univariate/BrentOptimizer.html].

[The Apache Software Foundation, 2016b] The Apache Software Foundation (2016b). Class NelderMeadSimplex. [Retrieved August 31, 2018, from http://commons.apache.org/proper/commons-math/javadocs/api-3.6/org/apache/commons/math3/optim/nonlinear/scalar/noderiv/NelderMeadSimplex.html].

[The Apache Software Foundation, 2016c] The Apache Software Foundation (2016c). Class SobolSequenceGenerator. [Retrieved August 31, 2018, from http://commons.apache.org/proper/commons-math/javadocs/api-3.3/org/apache/commons/math3/random/SobolSequenceGenerator.html].

[The Apache Software Foundation, 2016d] The Apache Software Foundation (2016d). Commons math: The apache commons mathematics library. [Retrieved August 31, 2018, from http://commons.apache.org/proper/commons-math/javadocs/api-3.6.1/index.html].

A

Appendix

A.1 Convergence Comparison Experiment

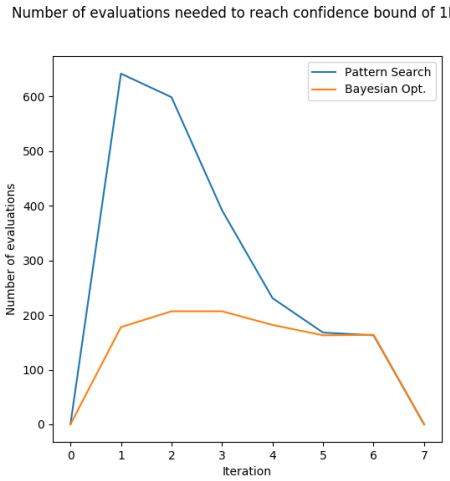
To further show the complete results of the experiment, all plot generated for the Convergence Comparison experiment are displayed here. The plots belonging to the same main domain (Nearest-Bid, Proxy, Proportional or Quadratic) are displayed in a 2×2 grid.

A.1.1 Evaluations needed per Iteration

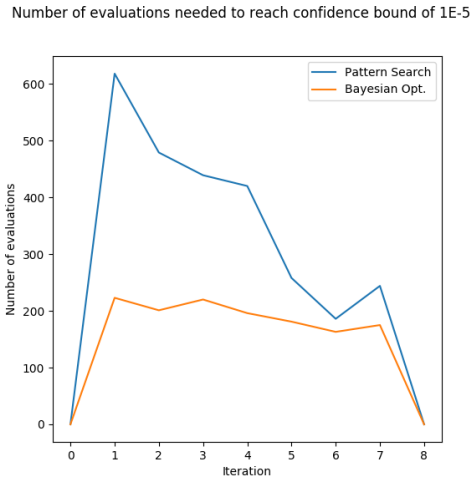
This group of plots displays how many evaluations were needed to reach a confidence region of 10^{-5} around a precalculated ground-truth value. For each iteration, indicated on the x-axis, we display the number of evaluations on the y-axis. For the number of evaluations we show the sum over all the 40 grid points.

BO as strategy-defining algorithm

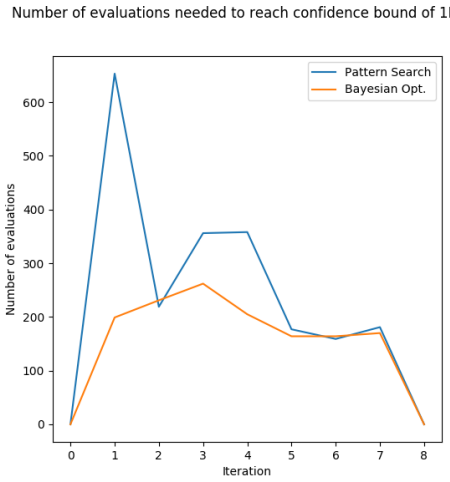
For these plots the BNE algorithm uses the strategies produced by the BO to continue with in the next iteration.



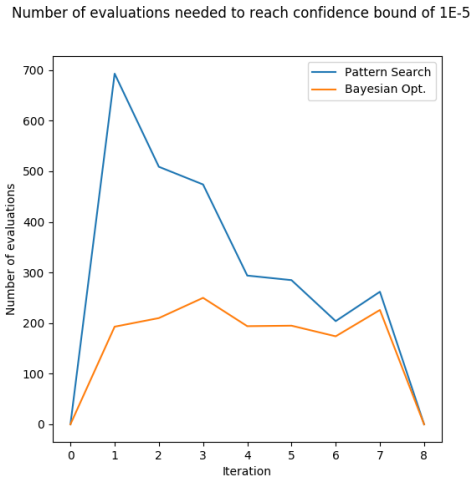
$\alpha = 1, \gamma = 0$



$\alpha = 1, \gamma = 0.5$

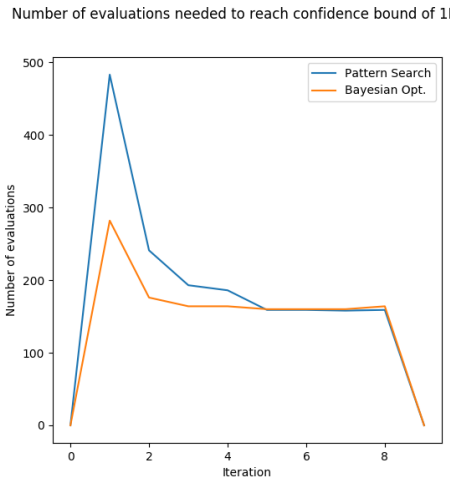


$\alpha = 2, \gamma = 0$

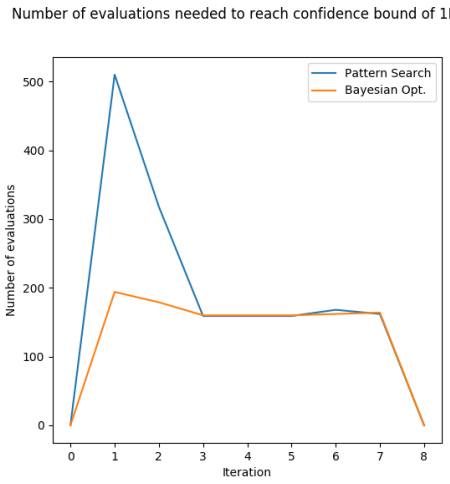


$\alpha = 2, \gamma = 0.5$

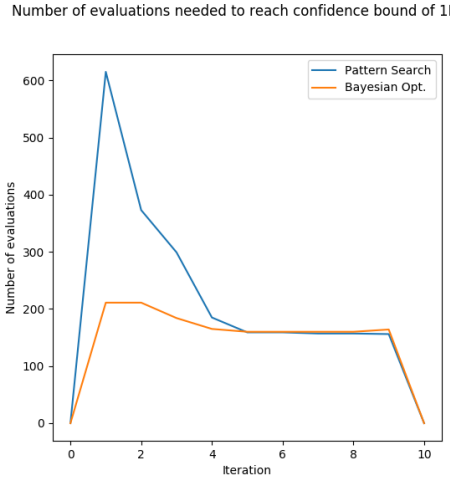
Figure A.1: **Evaluations per iteration for Nearest-Bid, BO is strategy defining**



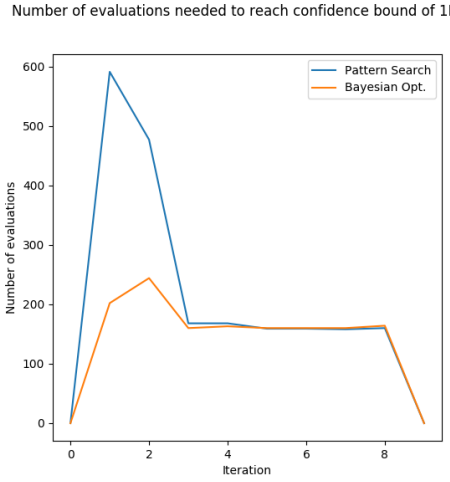
$\alpha = 1, \gamma = 0$



$\alpha = 1, \gamma = 0.5$

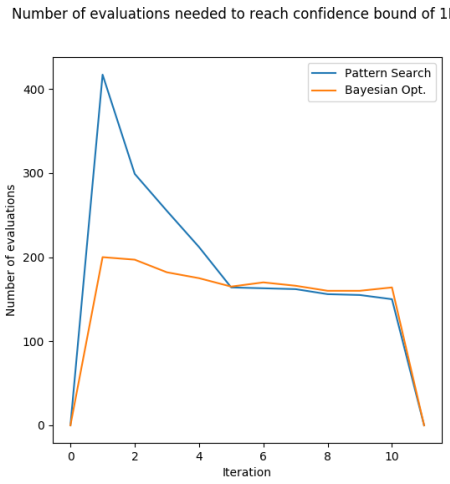


$\alpha = 2, \gamma = 0$

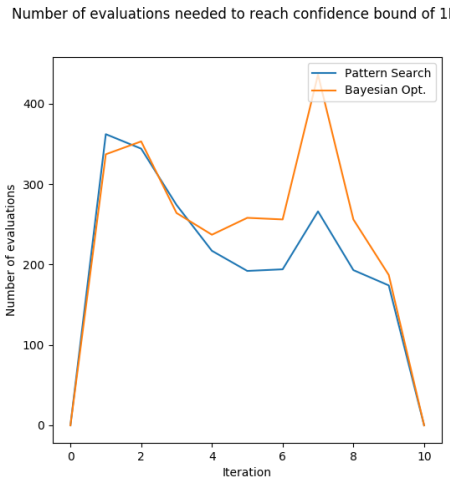


$\alpha = 2, \gamma = 0.5$

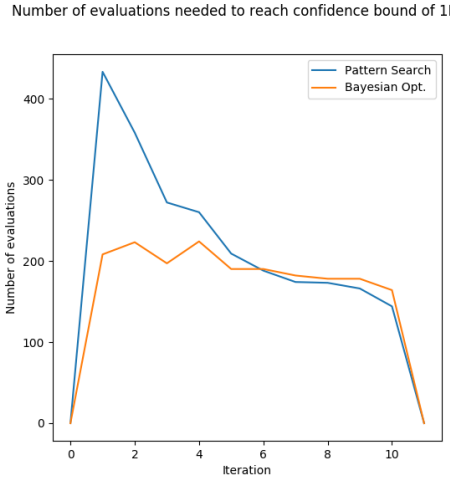
Figure A.2: **Evaluations per iteration for Proportional, BO is strategy defining**



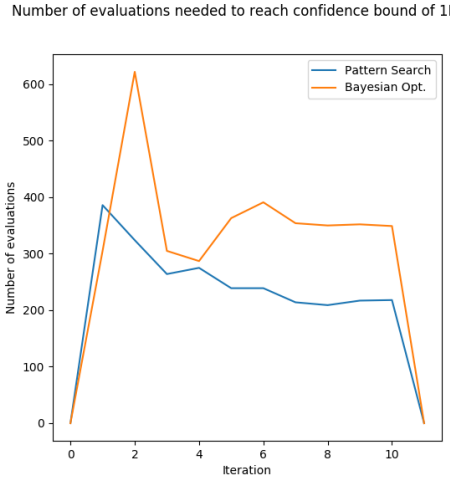
$\alpha = 1, \gamma = 0$



$\alpha = 1, \gamma = 0.5$

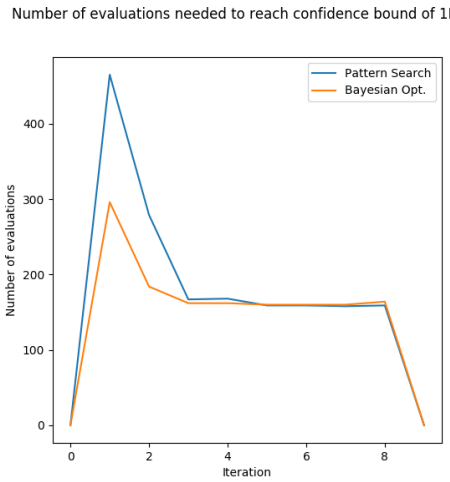


$\alpha = 2, \gamma = 0$

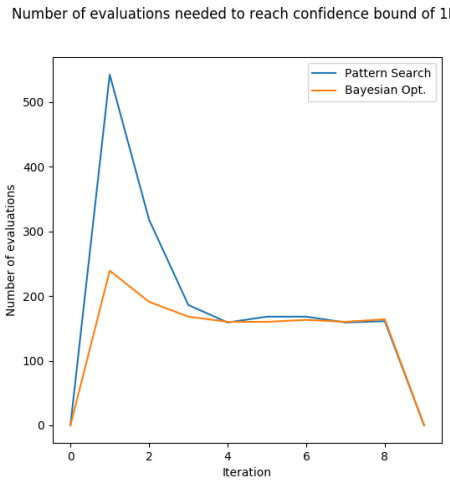


$\alpha = 2, \gamma = 0.5$

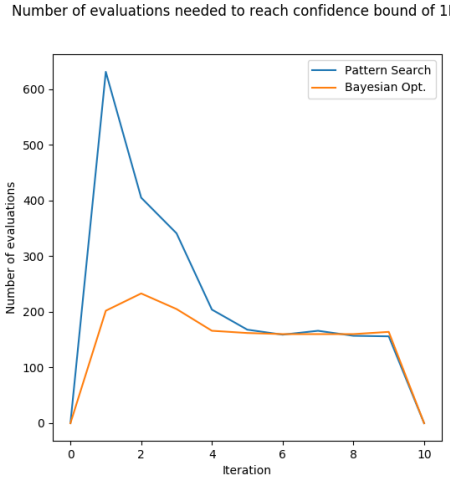
Figure A.3: **Evaluations per iteration for Proxy, BO is strategy defining**



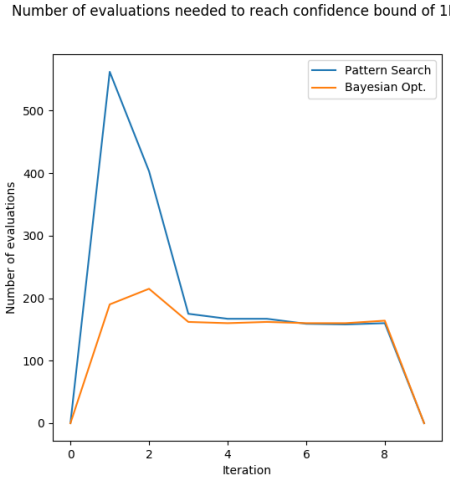
$\alpha = 1, \gamma = 0$



$\alpha = 1, \gamma = 0.5$



$\alpha = 2, \gamma = 0$

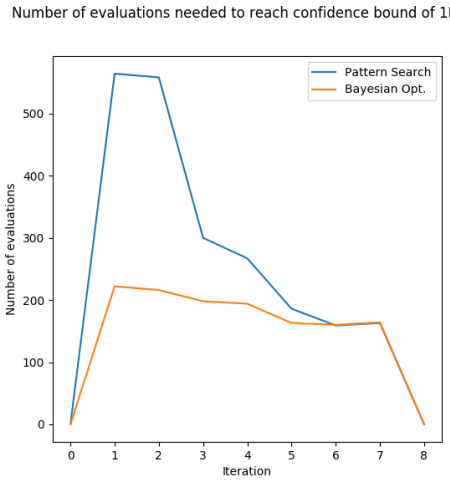


$\alpha = 2, \gamma = 0.5$

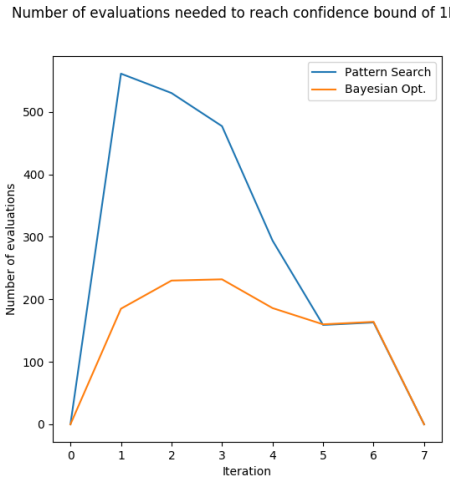
Figure A.4: **Evaluations per iteration for Quadratic, BO is strategy defining**

PS as strategy-defining algorithm

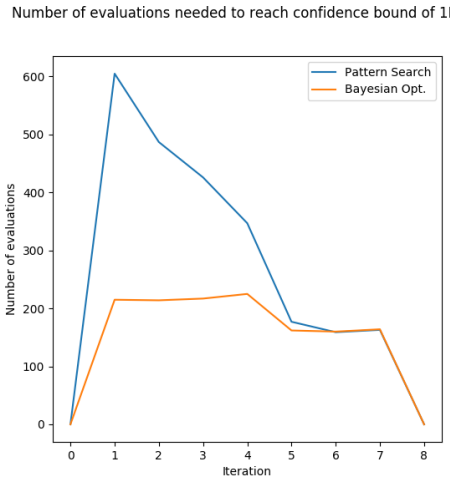
For these plots the BNE algorithm uses the strategies produced by the PS to continue with in the next iteration.



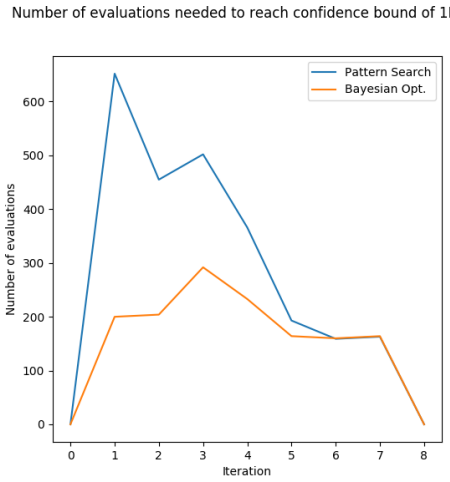
$\alpha = 1, \gamma = 0$



$\alpha = 1, \gamma = 0.5$

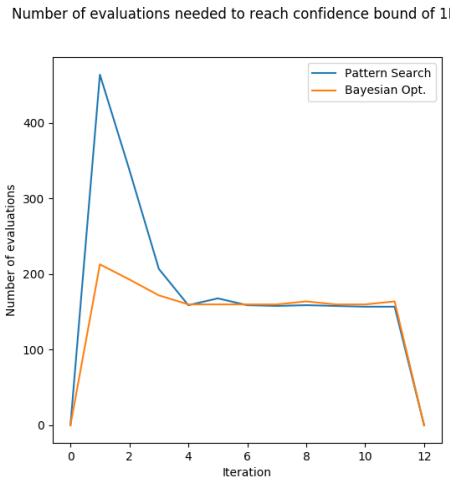


$\alpha = 2, \gamma = 0$

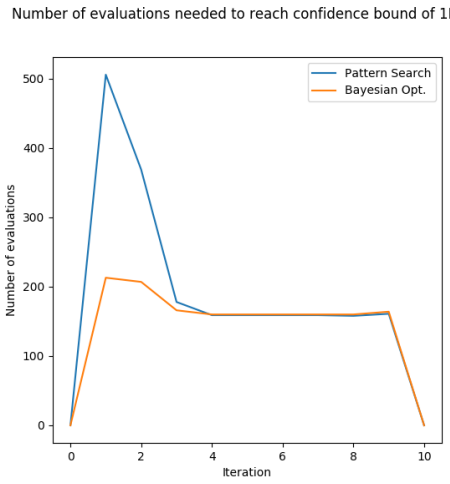


$\alpha = 2, \gamma = 0.5$

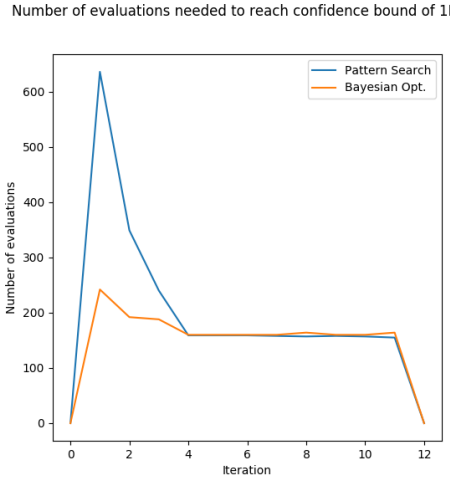
Figure A.5: **Evaluations per iteration for Nearest-Bid, PS is strategy defining**



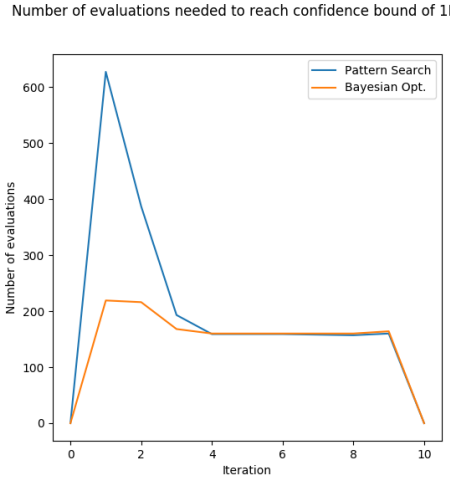
$\alpha = 1, \gamma = 0$



$\alpha = 1, \gamma = 0.5$

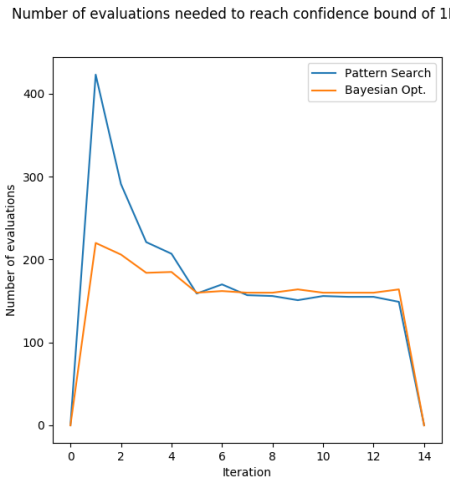


$\alpha = 2, \gamma = 0$

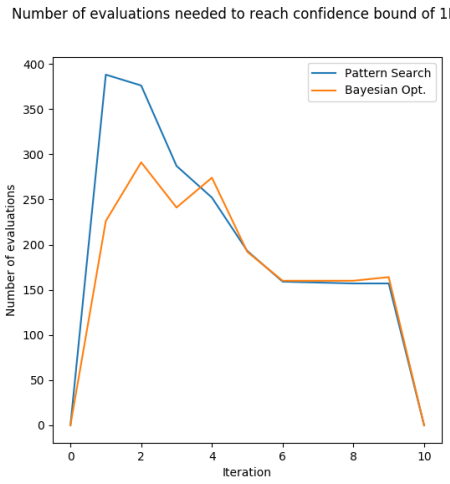


$\alpha = 2, \gamma = 0.5$

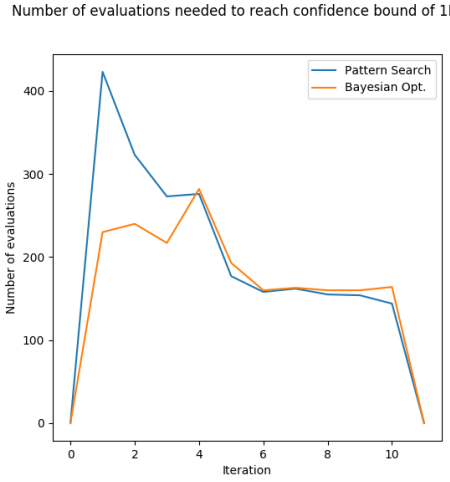
Figure A.6: **Evaluations per iteration for Proportional, PS is strategy defining**



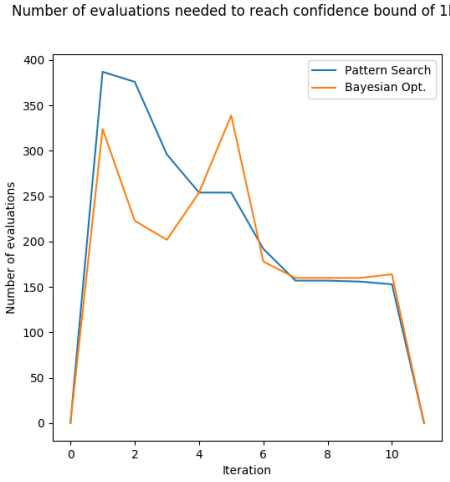
$\alpha = 1, \gamma = 0$



$\alpha = 1, \gamma = 0.5$

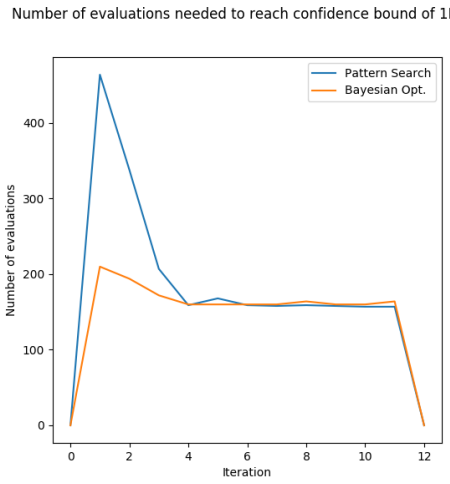


$\alpha = 2, \gamma = 0$

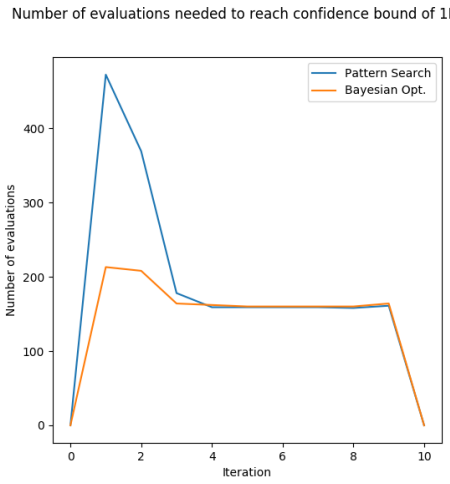


$\alpha = 2, \gamma = 0.5$

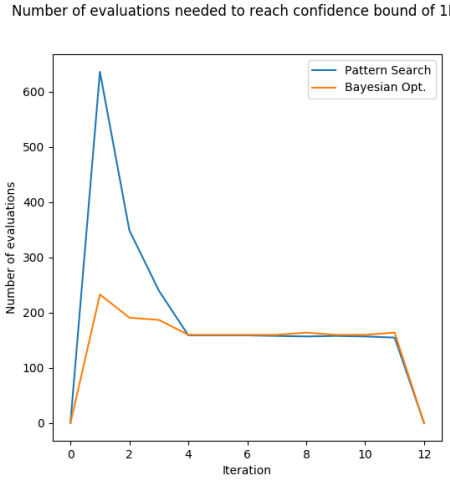
Figure A.7: **Evaluations per iteration for Proxy, PS is strategy defining**



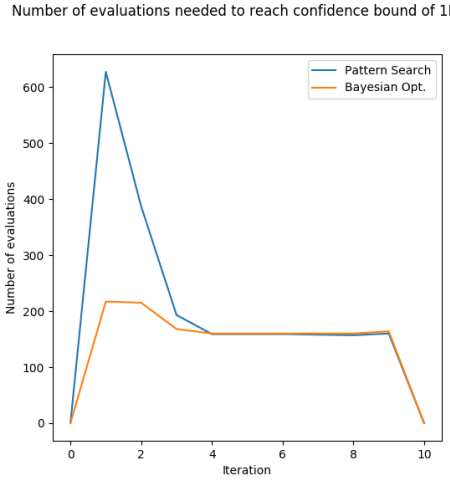
$\alpha = 1, \gamma = 0$



$\alpha = 1, \gamma = 0.5$



$\alpha = 2, \gamma = 0$



$\alpha = 2, \gamma = 0.5$

Figure A.8: **Evaluations per iteration for Quadratic, PS is strategy defining**

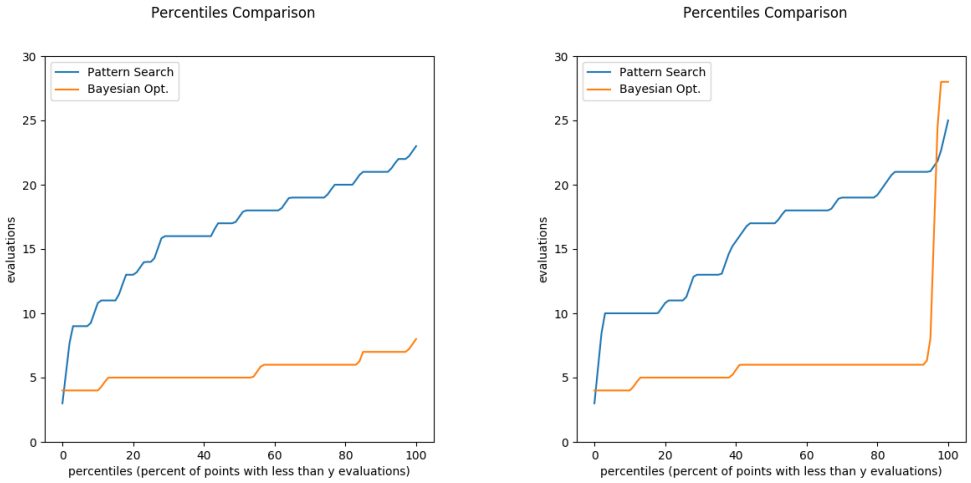
A.1.2 Percentiles

Since the maximum number of iterations that is reached by any point is almost equal for BO and PS, we also present a series of plots where we show how the required number of evaluations are distributed in percentiles. These plot can be interpreted the following way: Starting on the y-axis and selecting any arbitrary y-value y. Then on the x-axis we see how many percent of the 40 points need less than y evaluations to reach a region of 10^{-5} around the pre-calculated ground-truth value. These plots are particularly useful when we want to define a limit of evaluations the algorithms are allowed to use. If we say that we allow the algorithms to use y evaluations, then we can read from the plot that x percent of the value will be found correctly[1].

[1] With respect to the granted threshold of 10^{-5}.

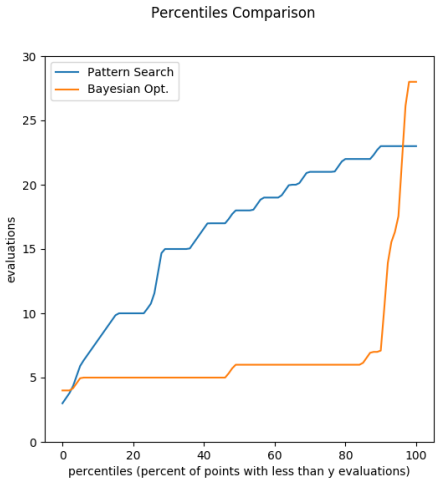
BO as strategy-defining algorithm

For these plots the CA-BNE algorithm uses the strategies produced by the BO to continue with in the next iteration.

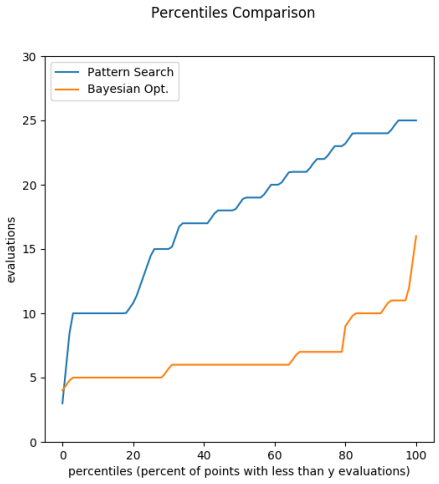


$\alpha = 1, \gamma = 0$

$\alpha = 1, \gamma = 0.5$

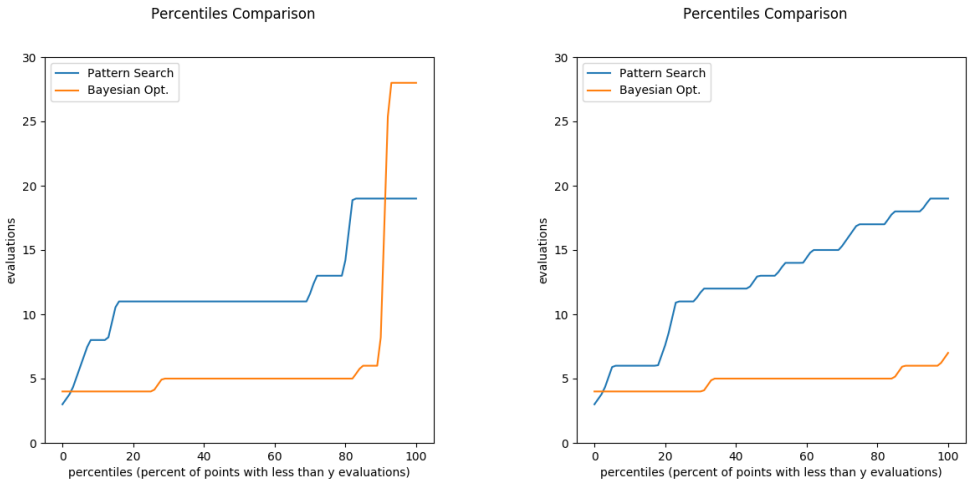


$\alpha = 2, \gamma = 0$



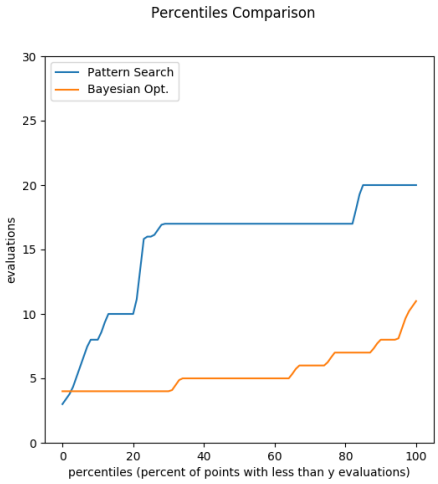
$\alpha = 2, \gamma = 0.5$

Figure A.9: **Percentiles for Nearest-Bid, BO is strategy defining**

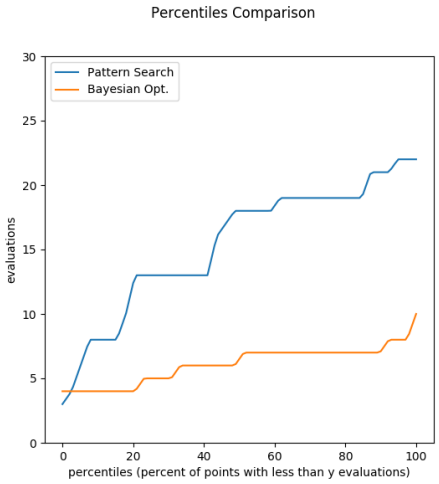


$\alpha = 1, \gamma = 0$

$\alpha = 1, \gamma = 0.5$

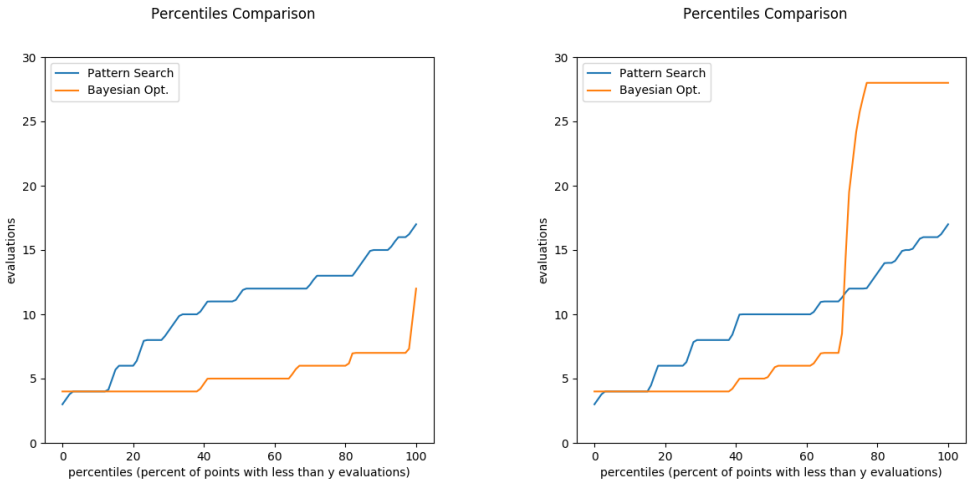


$\alpha = 2, \gamma = 0$



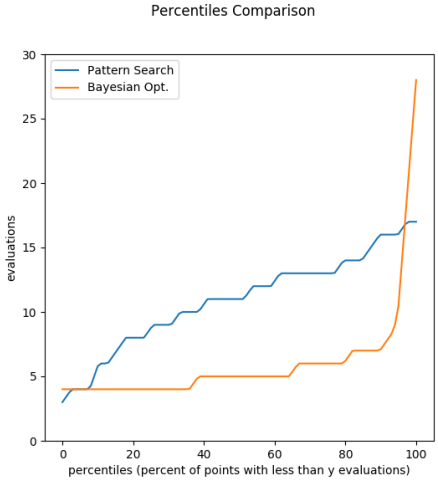
$\alpha = 2, \gamma = 0.5$

Figure A.10: **Percentiles for Proportional, BO is strategy defining**

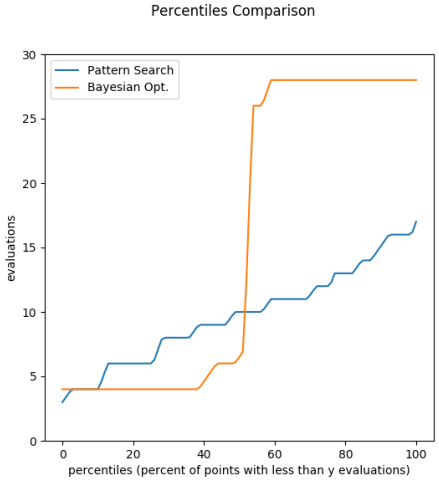


$\alpha = 1, \gamma = 0$

$\alpha = 1, \gamma = 0.5$

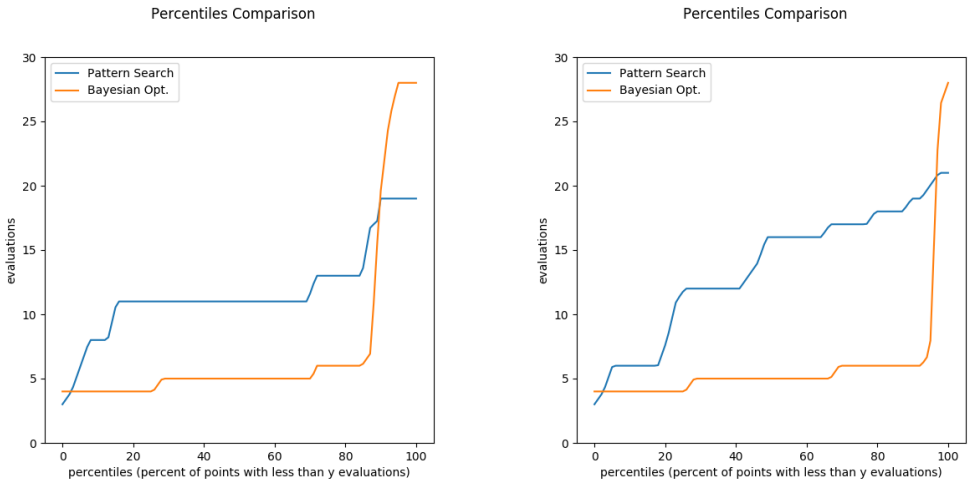


$\alpha = 2, \gamma = 0$



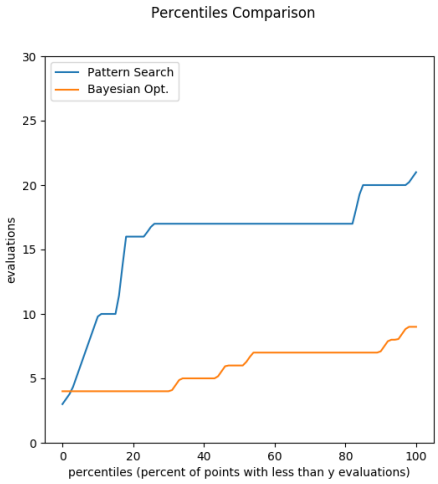
$\alpha = 2, \gamma = 0.5$

Figure A.11: **Percentiles for Proxy, BO is strategy defining**

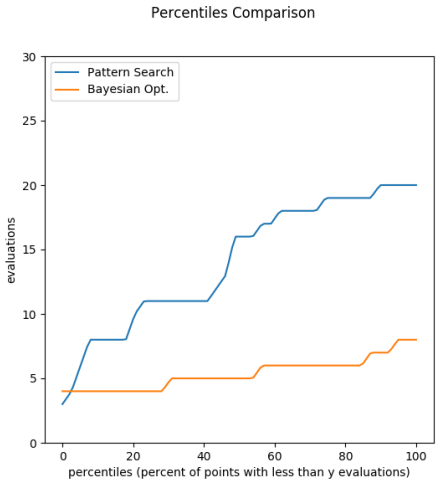


$\alpha = 1, \gamma = 0$

$\alpha = 1, \gamma = 0.5$



$\alpha = 2, \gamma = 0$

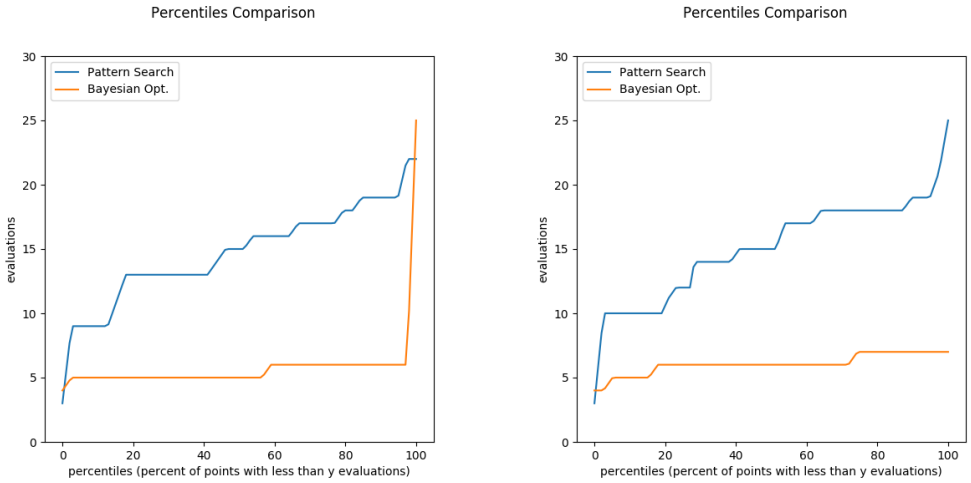


$\alpha = 2, \gamma = 0.5$

Figure A.12: **Percentiles for Quadratic, BO is strategy defining**

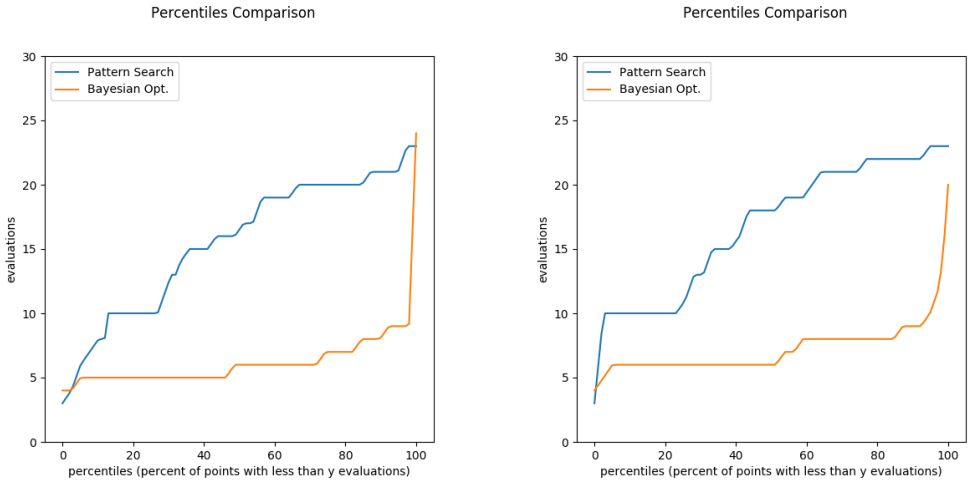
PS as strategy-defining algorithm

For these plots the CA-BNE algorithm uses the strategies produced by the PS to continue with in the next iteration.



$\alpha = 1, \gamma = 0$

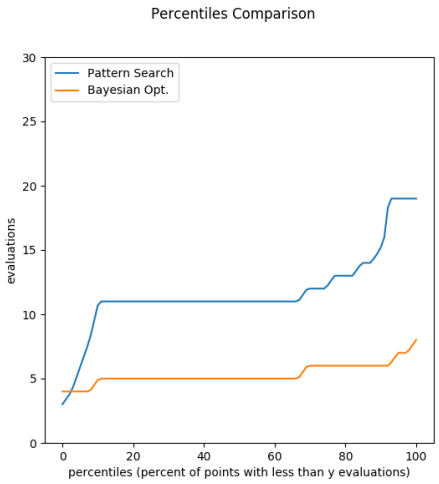
$\alpha = 1, \gamma = 0.5$



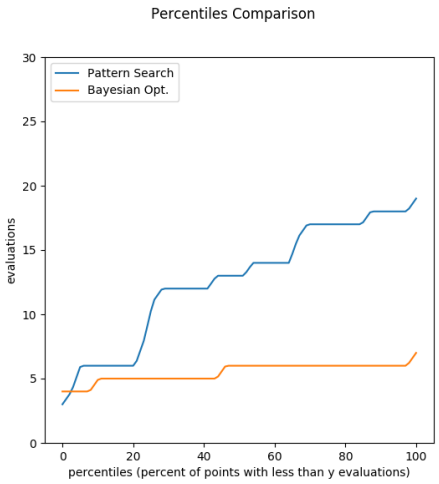
$\alpha = 2, \gamma = 0$

$\alpha = 2, \gamma = 0.5$

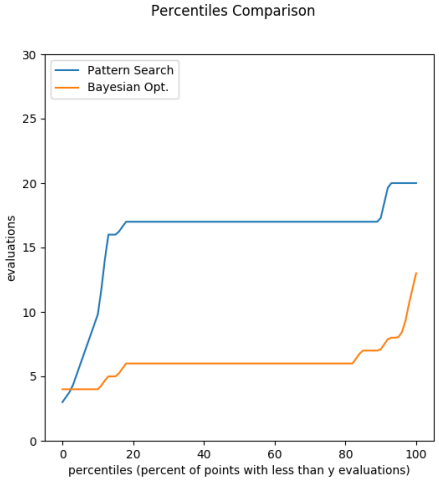
Figure A.13: **Percentiles for Nearest-Bid, PS is strategy defining**



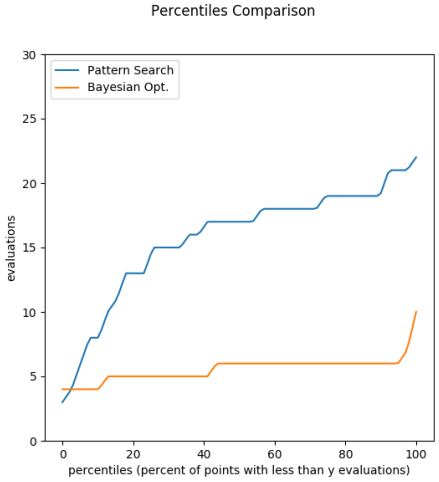
$\alpha = 1, \gamma = 0$



$\alpha = 1, \gamma = 0.5$

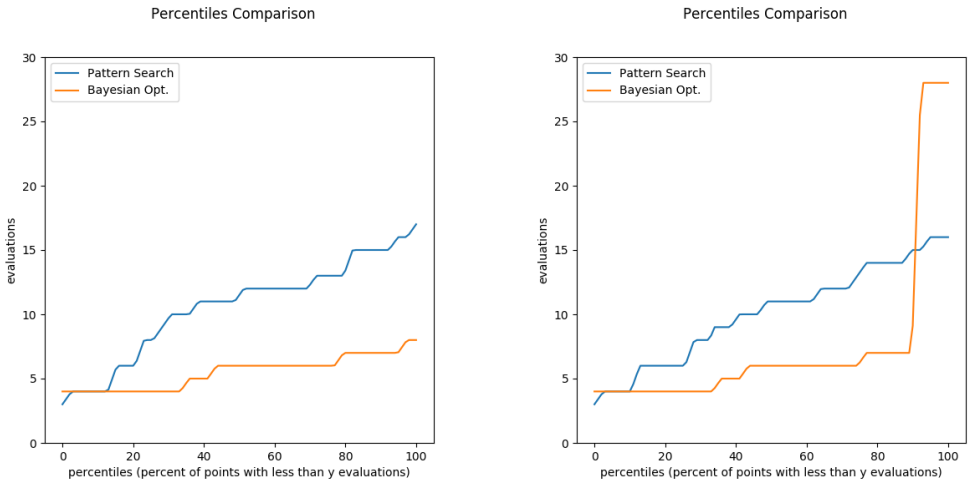


$\alpha = 2, \gamma = 0$



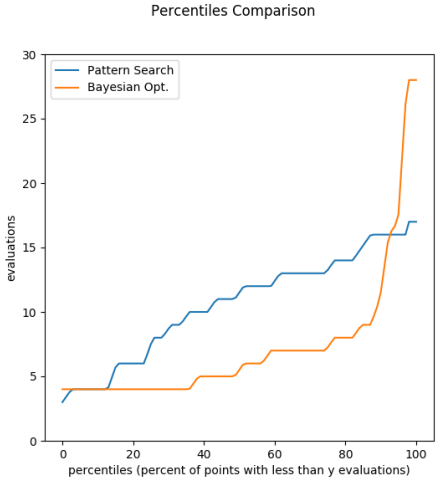
$\alpha = 2, \gamma = 0.5$

Figure A.14: **Percentiles for Proportional, PS is strategy defining**

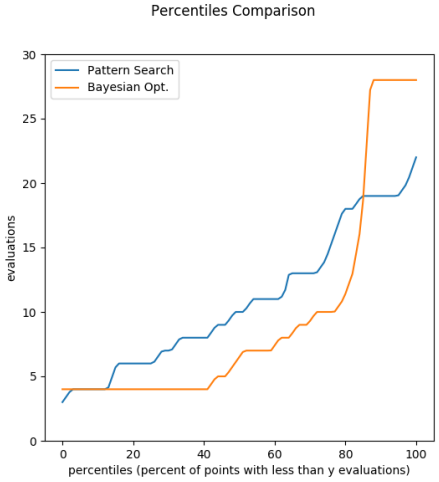


$\alpha = 1, \gamma = 0$

$\alpha = 1, \gamma = 0.5$

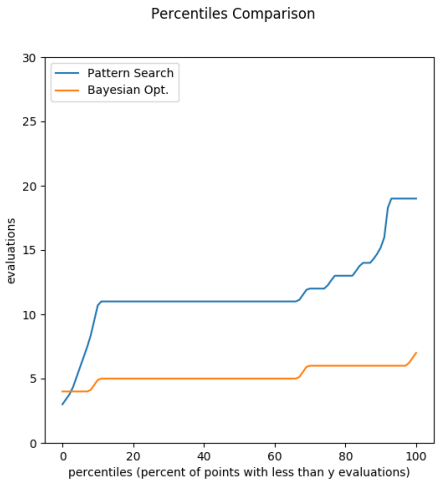


$\alpha = 2, \gamma = 0$

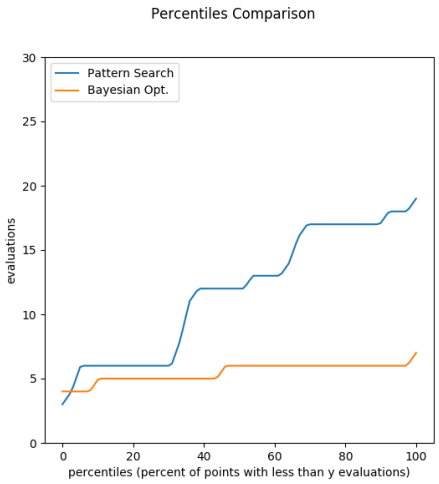


$\alpha = 2, \gamma = 0.5$

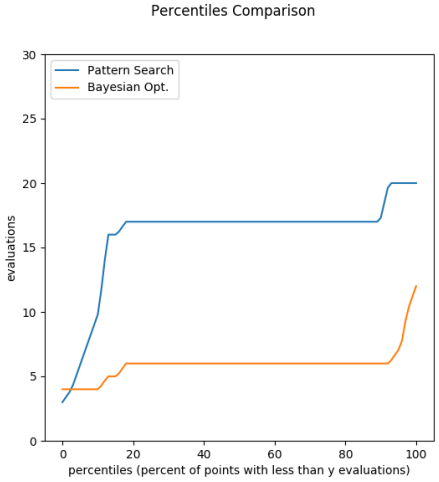
Figure A.15: **Percentiles for Proxy, PS is strategy defining**



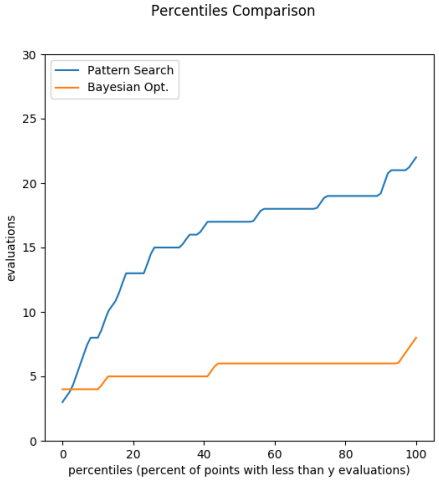
$\alpha = 1, \gamma = 0$



$\alpha = 1, \gamma = 0.5$



$\alpha = 2, \gamma = 0$



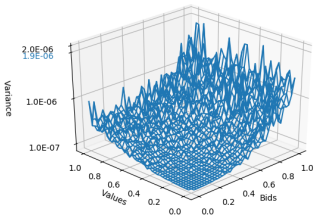
$\alpha = 2, \gamma = 0.5$

Figure A.16: **Percentiles for Quadratic, PS is strategy defining**

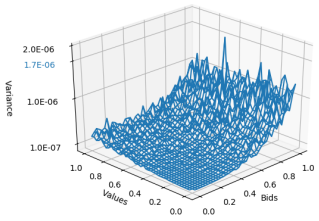
A.2 Variance Experiment

To be able to asses how the variance for different bid/value combinations evolves, we plot it using a mesh-grid figure. On the z-axis the variance is displayed. Looking at the plot we get an impression of how the shape of the variance curve looks like. To see that shape clearly the plots are in three different scales. The default scale of is $2 * 10^{-5}$ and the other scales are two times and three times the default scale. For each of the 16 domains there is a plot for each iteration of the CA-BNE algorithm, so for each new best response calculation. We see that with increasing iteration index the variance slowly decreases. The shape of the curve, however, stays the same.

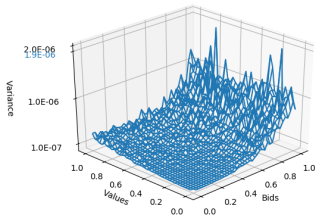
Nearest Bid



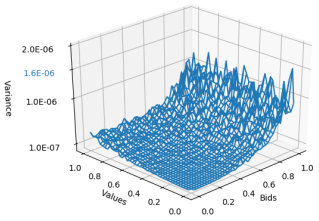
iteration 0



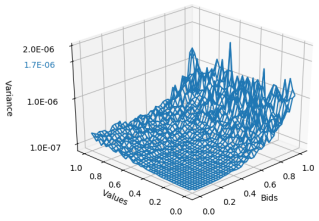
iteration 1



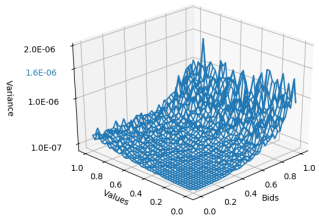
iteration 2



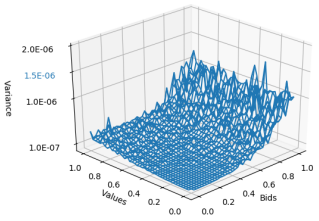
iteration 3



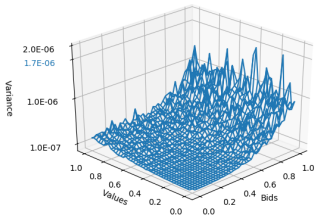
iteration 4



iteration 5



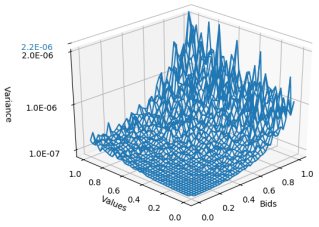
iteration 6



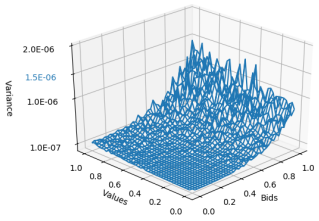
iteration 7

Figure A.17: **Variance of the Nearest-Bid mechanism,**$\alpha = 1$ **and** $\gamma = 0$, scale: 1x

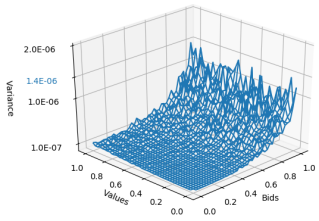
Nearest Bid



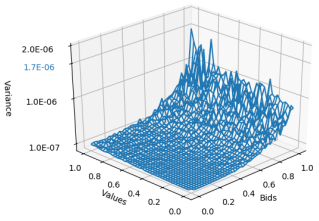
iteration 0



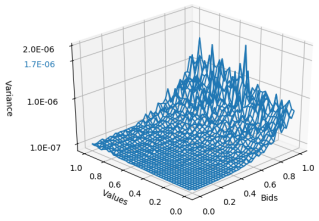
iteration 1



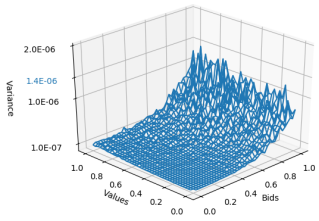
iteration 2



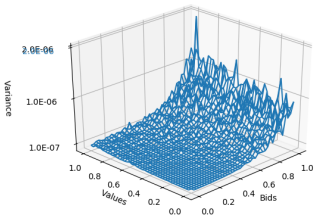
iteration 3



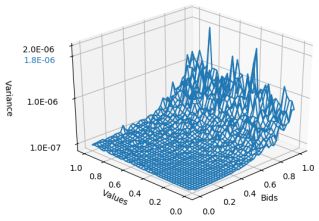
iteration 4



iteration 5



iteration 6

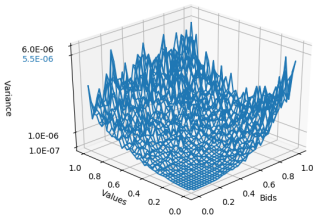


iteration 7

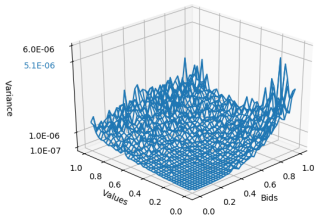
Figure A.18: **Variance of the Nearest-Bid mechanism,** $\alpha = 1$ **and** $\gamma = 0.5$, scale: 1x

85

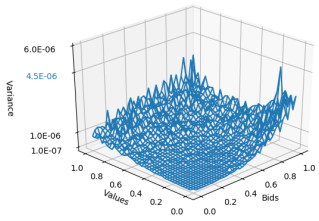
Nearest Bid



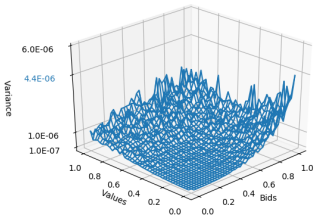
iteration 0



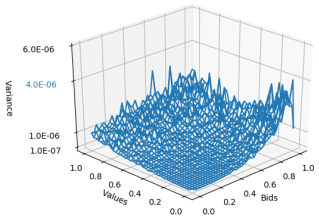
iteration 1



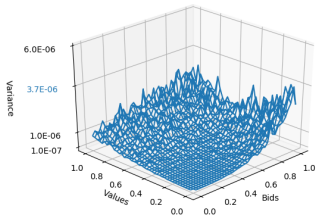
iteration 2



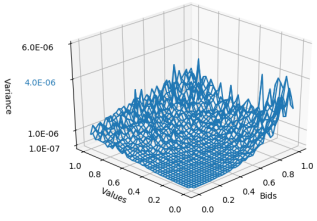
iteration 3



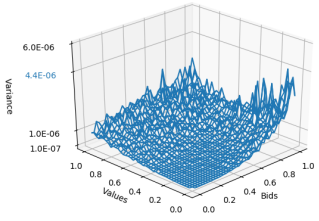
iteration 4



iteration 5



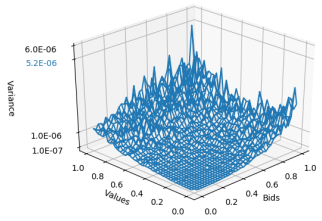
iteration 6



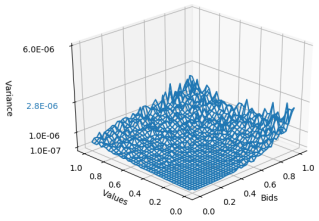
iteration 7

Figure A.19: **Variance of the Nearest-Bid mechanism,** $\alpha = 2$ **and** $\gamma = 0$, scale: 3x

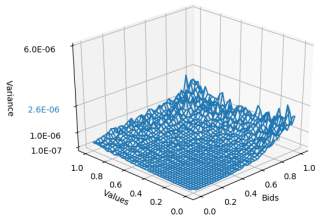
Nearest Bid



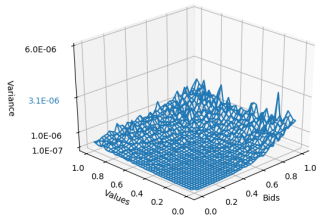
iteration 0



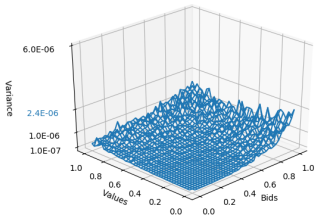
iteration 1



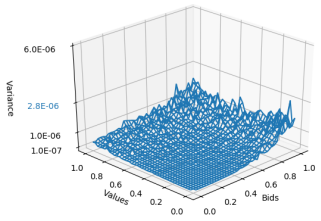
iteration 2



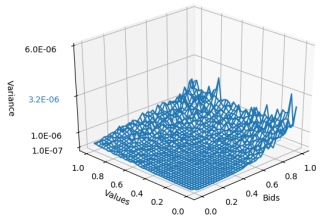
iteration 3



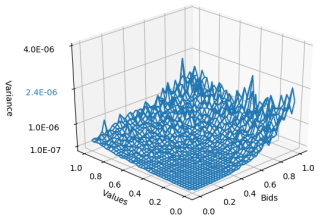
iteration 4



iteration 5



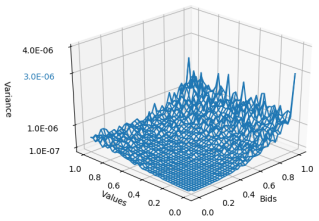
iteration 6



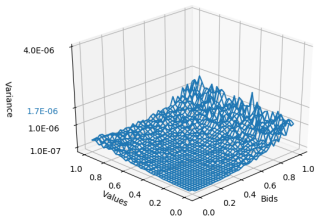
iteration 7

Figure A.20: **Variance of the Nearest-Bid mechanism,** $\alpha = 2$ **and** $\gamma = 0.5$, scale: 3x

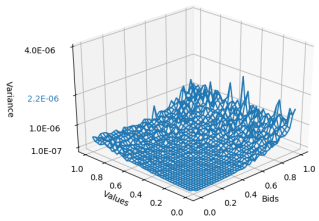
Proportional



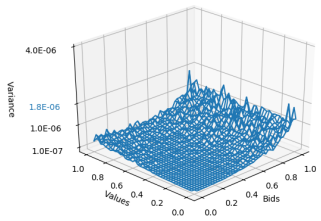
iteration 0



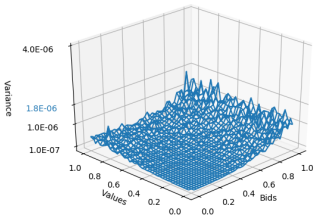
iteration 1



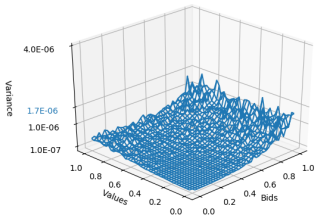
iteration 2



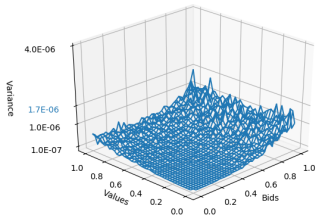
iteration 3



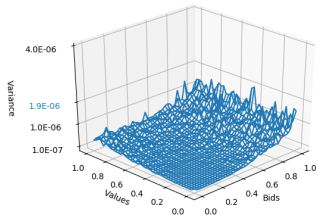
iteration 4



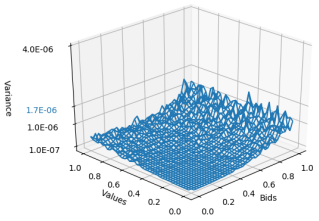
iteration 5



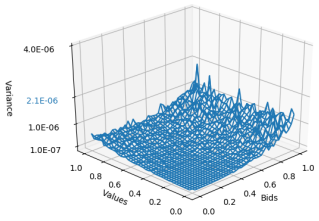
iteration 6



iteration 7



iteration 8



iteration 9

Figure A.21: **Variance of the Proportional mechanism,** $\alpha = 1$ **and** $\gamma = 0$, scale: 2x

Figure A.22: **Variance of the Proportional mechanism,** $\alpha = 1$ **and** $\gamma = 0.5$, scale: 2x

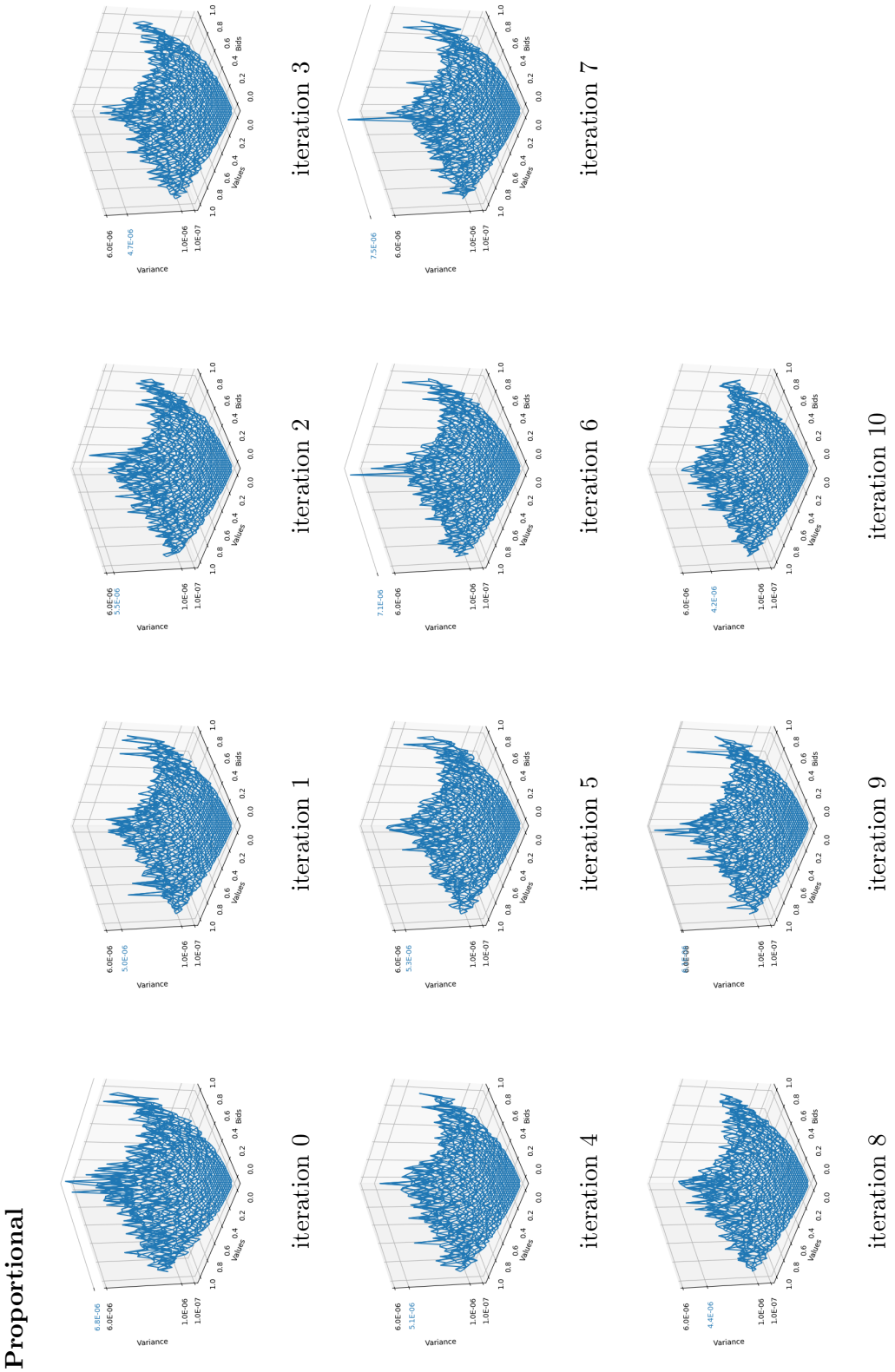


Figure A.23: **Variance of the Proportional mechanism, $\alpha = 2$ and $\gamma = 0$,** scale: 3x

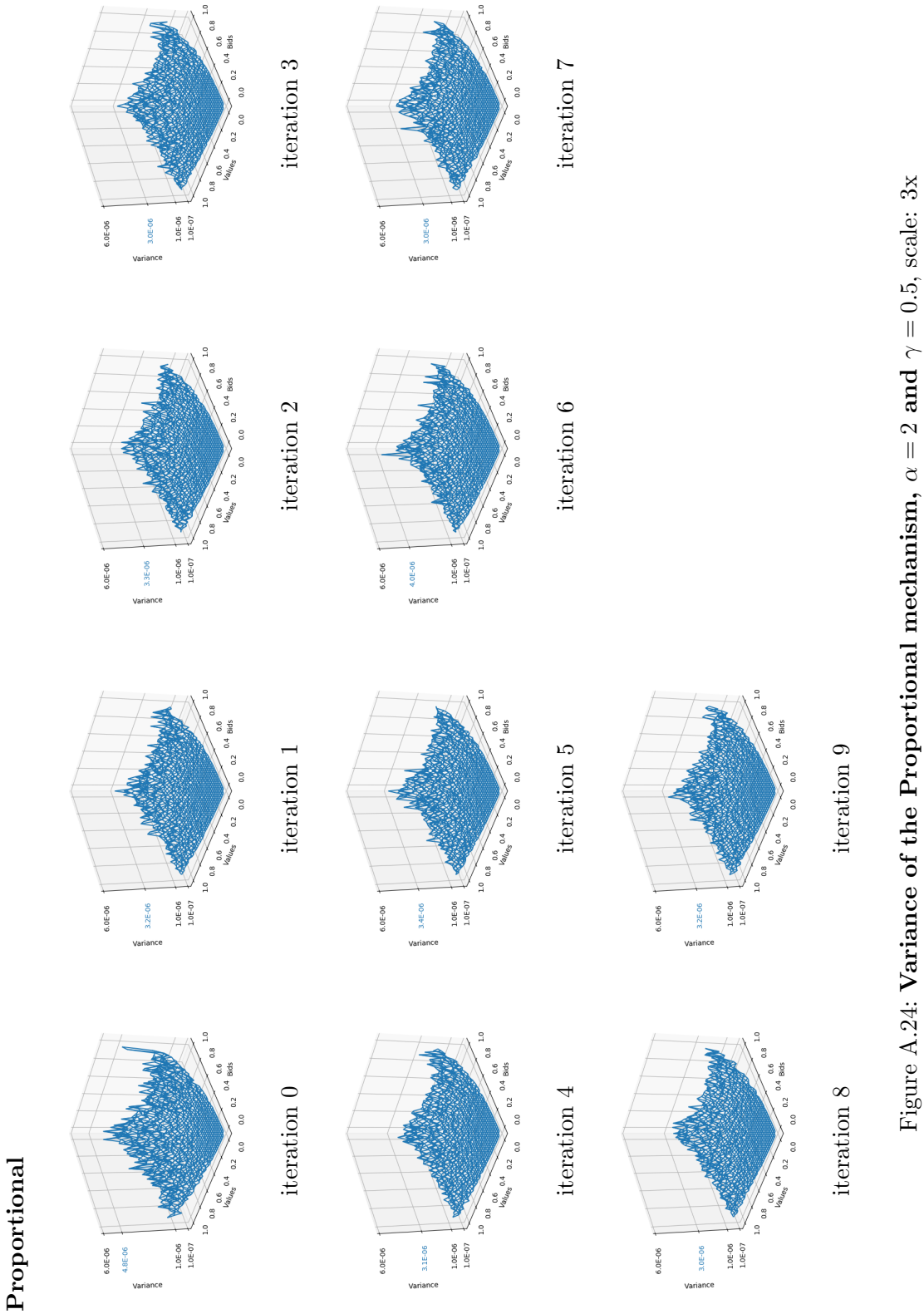


Figure A.24: **Variance of the Proportional mechanism,** $\alpha = 2$ **and** $\gamma = 0.5$, scale: 3x

Figure A.25: **Variance of the Proxy mechanism,** $\alpha = 1$ **and** $\gamma = 0$, scale: 2x

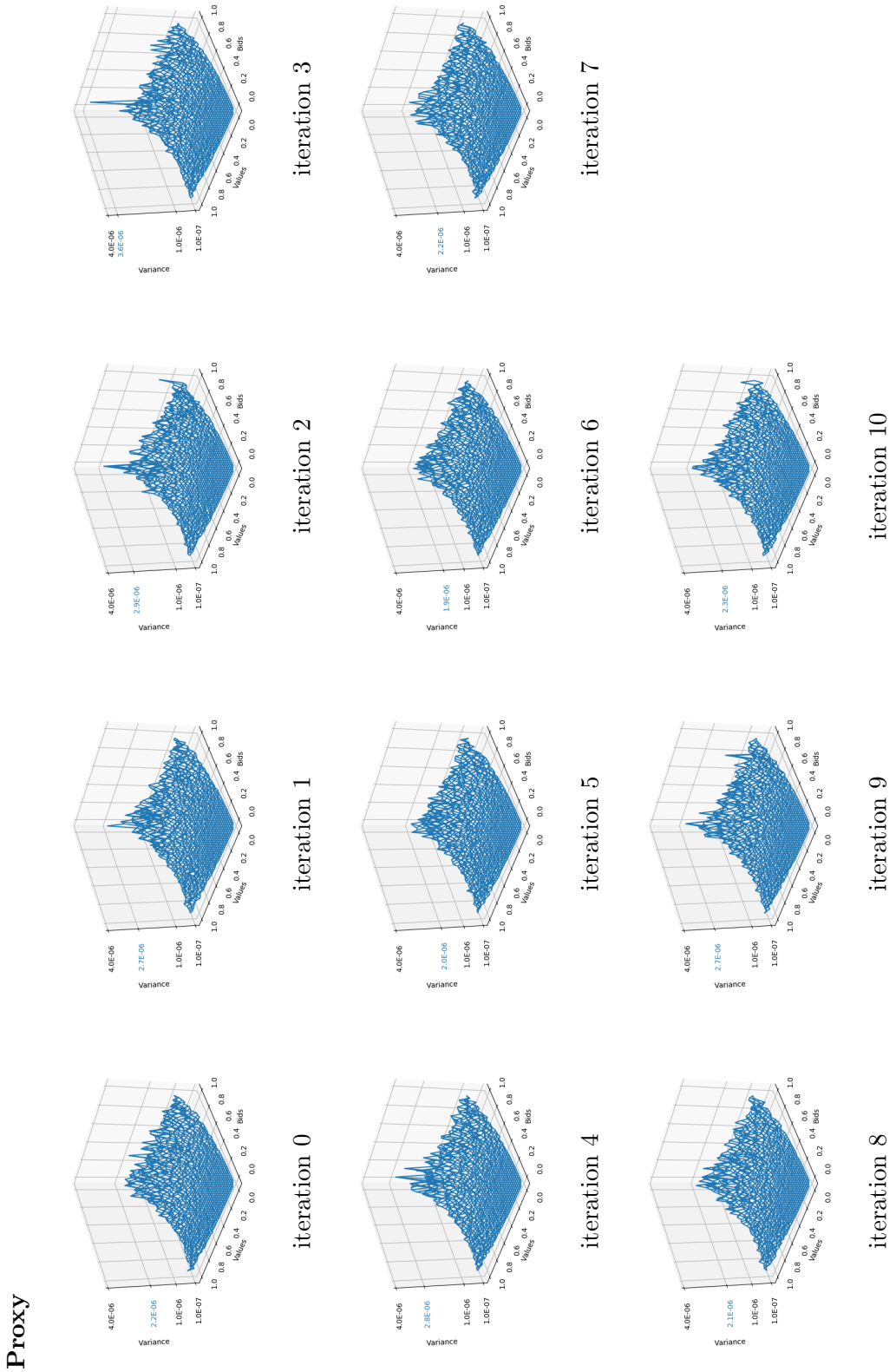


Figure A.26: **Variance of the Proxy mechanism,** $\alpha = 1$ **and** $\gamma = 0.5$, scale: 2x

Proxy

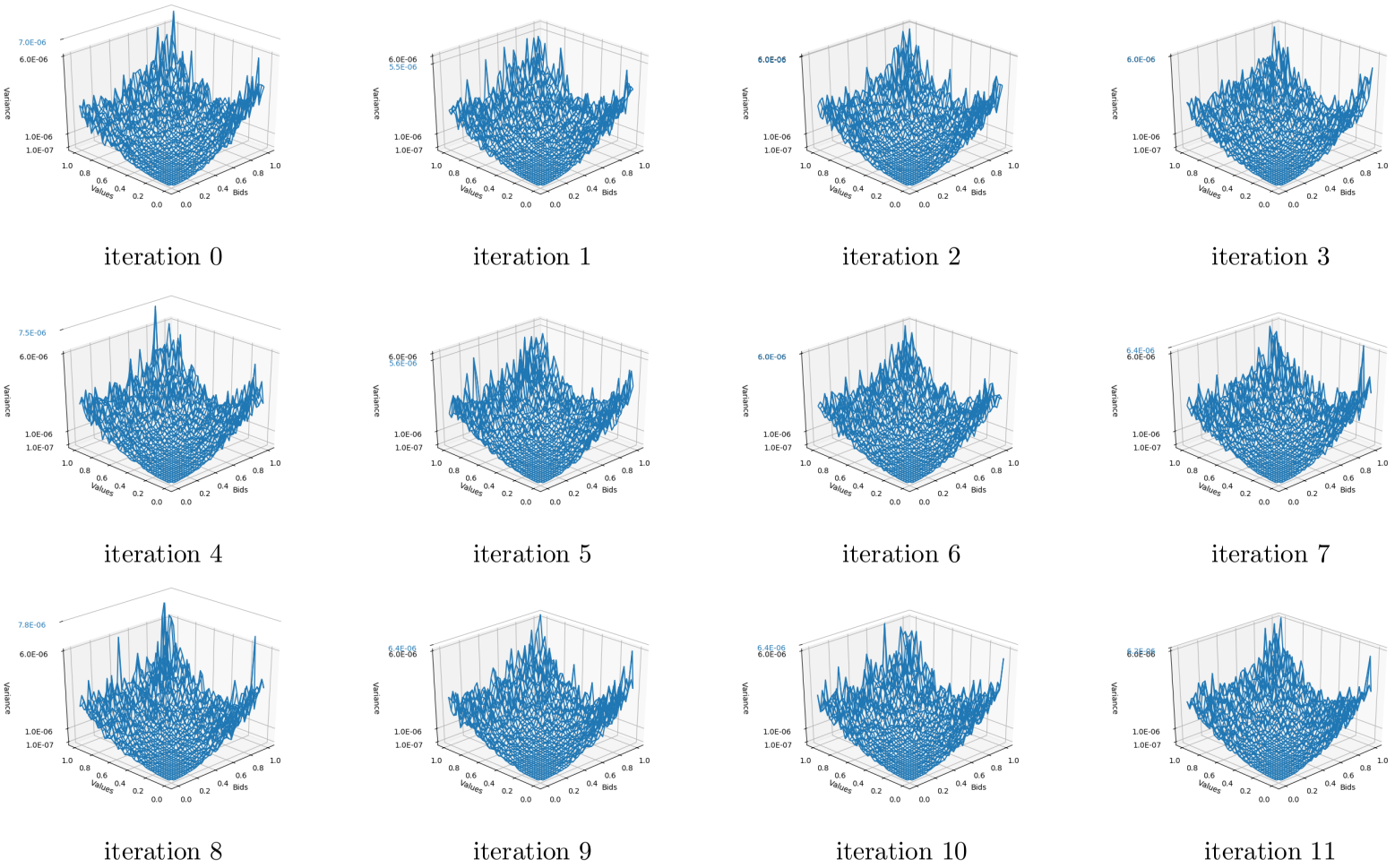


Figure A.27: **Variance of the Proxy mechanism,** $\alpha = 2$ **and** $\gamma = 0$, scale: 3x

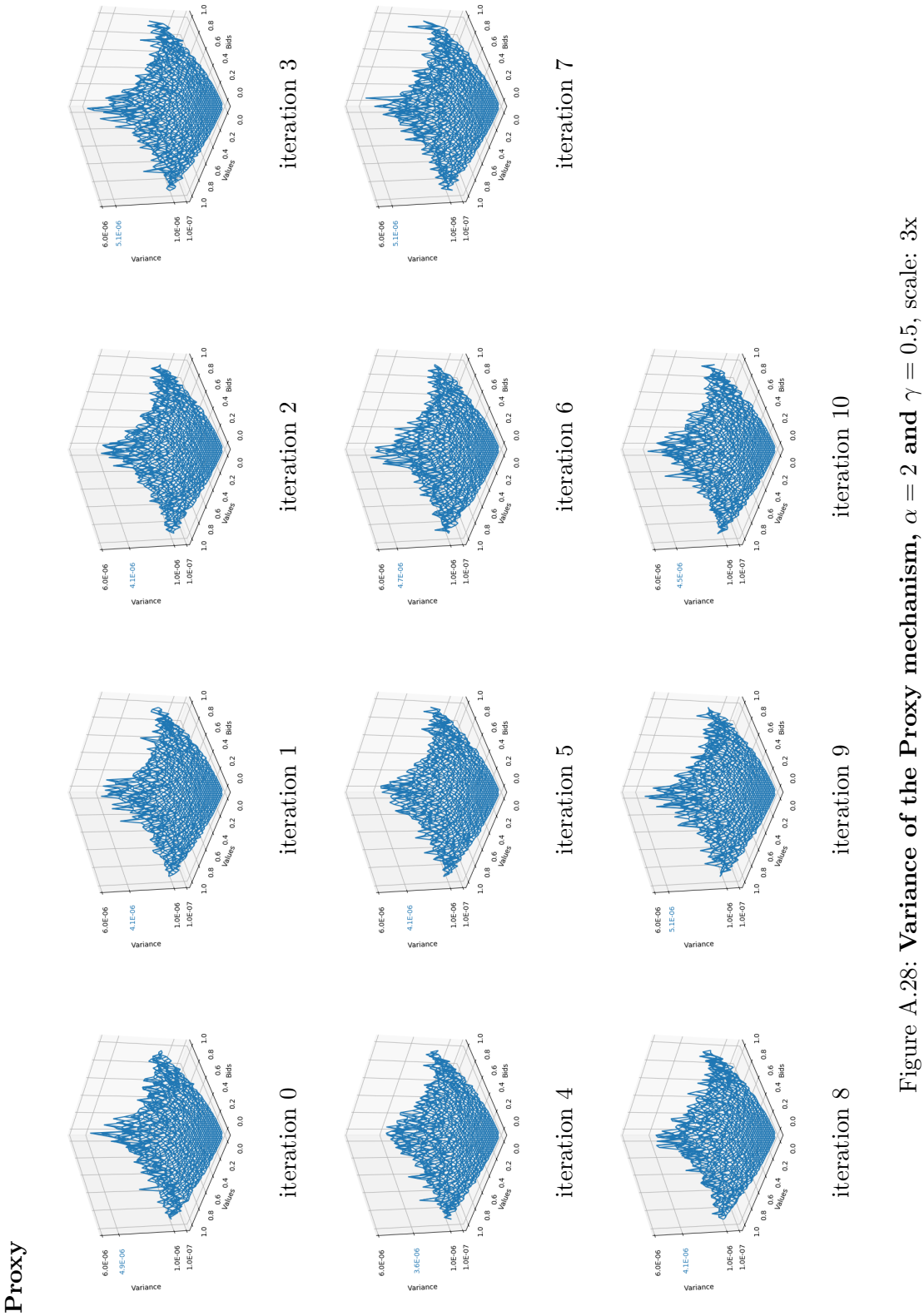
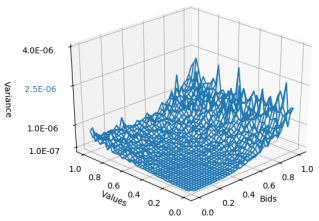
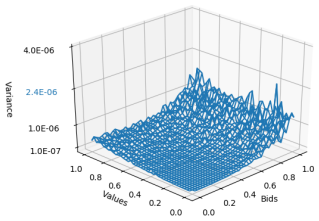


Figure A.28: **Variance of the Proxy mechanism,** $\alpha = 2$ **and** $\gamma = 0.5$, scale: 3x

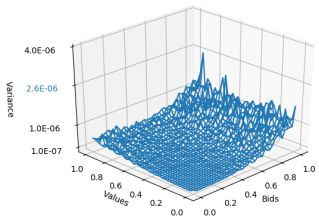
Quadratic



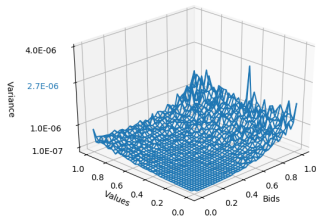
iteration 0



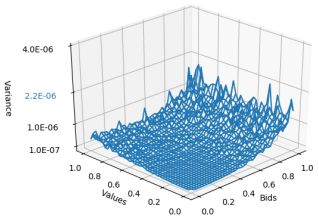
iteration 1



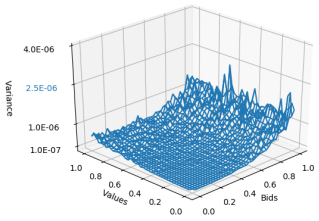
iteration 2



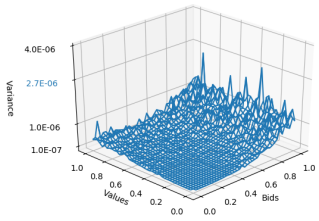
iteration 3



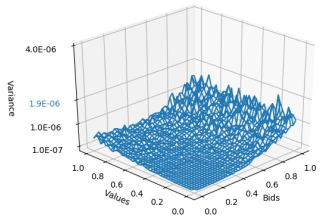
iteration 4



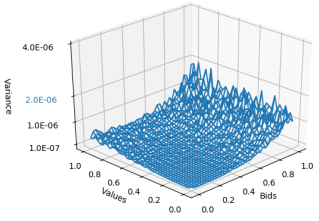
iteration 5



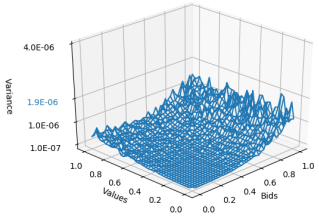
iteration 6



iteration 7



iteration 8



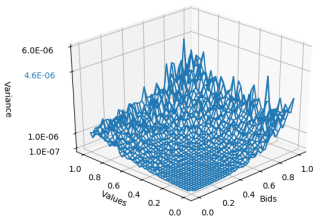
iteration 9

Figure A.29: **Variance of the Quadratic mechanism,** $\alpha = 1$ **and** $\gamma = 0$, scale: 2x

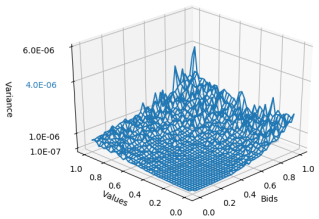
Figure A.30: **Variance of the Quadratic mechanism,** $\alpha = 1$ **and** $\gamma = 0.5$, scale: 2x

Figure A.31: **Variance of the Quadratic mechanism, $\alpha = 2$ and $\gamma = 0$,** scale: 3x

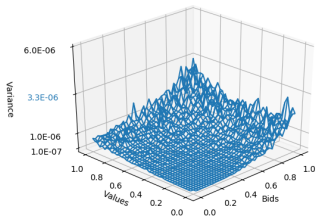
Quadratic



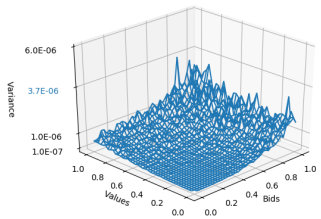
iteration 0



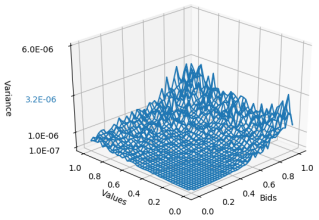
iteration 1



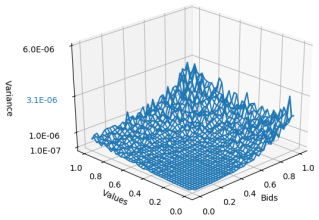
iteration 2



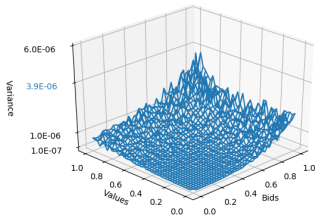
iteration 3



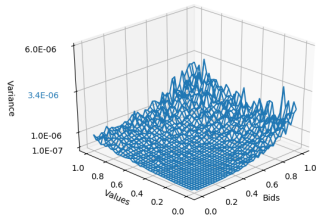
iteration 4



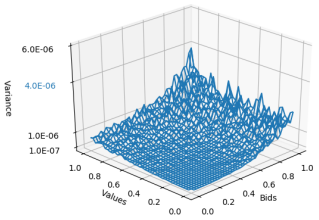
iteration 5



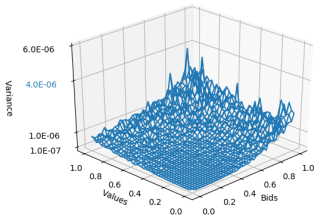
iteration 6



iteration 7



iteration 8



iteration 9

Figure A.32: **Variance of the Quadratic mechanism,** $\alpha = 2$ **and** $\gamma = 0.5$, scale: 3x

99

A.3 Bayesian Optimization Analysis

In the experiment to analyse the BO we want to find out whether the final strategies produced by the CA-BNE algorithm using BO are viable. Since the 16 domains we are using in this thesis do have an analytical solution we compare it against the final strategies produced using BO or PS.
Also, we want to look further into how the BO works and what steps are involved. For this we show the plots that picture the process of calculating mean and variance in the GP, maximizing the acquisition function and then evaluation acquisition-maximizing point.

A.3.1 Analytical Results Algorithms

The analytical results were derived by [Ausubel and Baranov, 2018]. However, in [Ausubel and Baranov, 2018] the analytical results are are only clear for $\alpha = 1$. The solutions for the more general case with any α and γ are not directly visible, but can be reconstructed. Therefore, the algorithms[2] to evaluate the analytical solution for all the 16 domains are presented here, so that it is clear what exact analytical solution were used in this thesis.
These algorithms are also conform with the corrections proposed by [Bosshard et al., 2017].

Algorithm 4: Analytical result Quadratic and Proportional

Data: α, γ, value v
if γ == *1* **then**
  k = $2/(2+\gamma)$;
  d = $(3k^2 - 2k\sqrt{3k-1})/(3k-2)$;
  return `max`$(0, k \cdot v - d)$;
else
  k = $2/(2+\gamma)$;
  s = `solve`$(x^{\alpha+1}/((1+\alpha)k^{\alpha}) - (3k \cdot x)/(3k-2) + (\alpha \cdot k)/(\alpha+1) + 1, 1, x)$
  return `max`$(0, k \cdot v - s)$;

[2]These analytical background for these algorithm are results from the works of [Ausubel and Baranov, 2018], the explicit formulations however, was kindly provided by Bosshard

Algorithm 5: Analytical result Proxy

Data: α, γ, value v

if γ == *1* **then**
  return v;
else if α==*1* **then**
  return $\texttt{max}(0,1 + (\log(\gamma + (1-\gamma)v)/(1-\gamma)))$
else if γ==*0* **then**
  if v==*0* **then**
    return 0;
  else
    return $\texttt{max}(0,1 + (\log(\gamma + (1-\gamma)v)/(1-\gamma)))$;
  end
  return $\texttt{max}(0,(v^{1-\alpha} - \alpha)/(1-\alpha))$;
else if α==*2* **then**
  $c = 1 - (1/\sqrt{\gamma(1-\gamma)})\texttt{arctan}(\sqrt{(1-\gamma)/\gamma})$;
  $\texttt{max}(0,(1/\sqrt{\gamma(1-\gamma)})\texttt{arctan}(\sqrt{(1-\gamma)/\gamma}v)+c)$
else
  invalid input
end

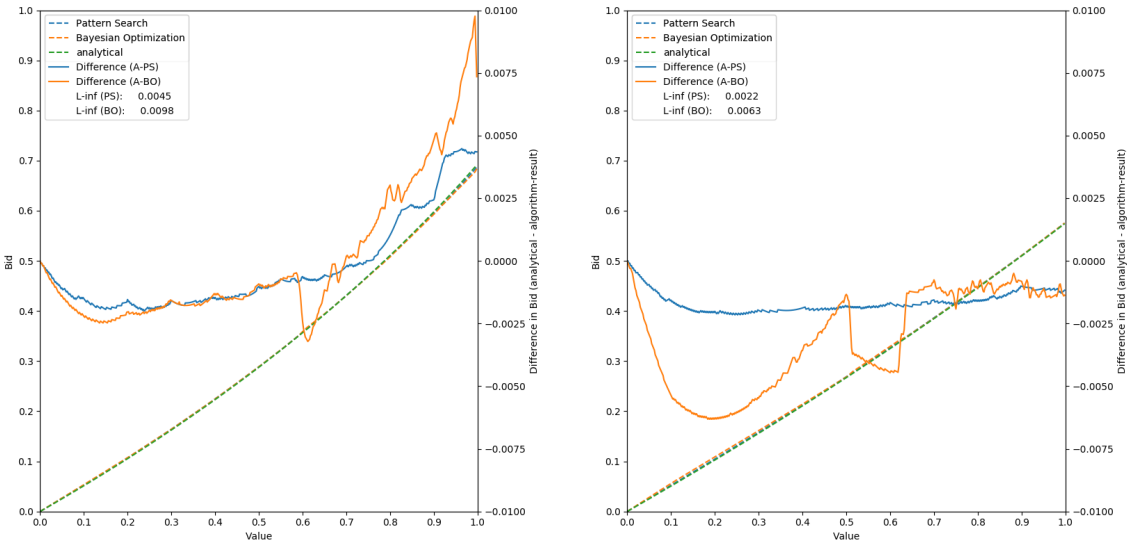
Algorithm 6: Analytical result Nearest-Bid

Data: α, γ, value v

if γ == *1* **then**
  return $v/2$;
else if α==*1* **then**
  return $(1/(1-\gamma))(log(2) - \log(2 - (1-\gamma)v))$
else if α==*2* **then**
  tmp1 = $\log(\sqrt{2/(1-\gamma)} + v)$;
  tmp2 = $\log(\sqrt{2/(1-\gamma)} - v)$;
  return $(1/\sqrt{8-8\gamma})(tmp1 - tmp2)$
else
  invalid input
end

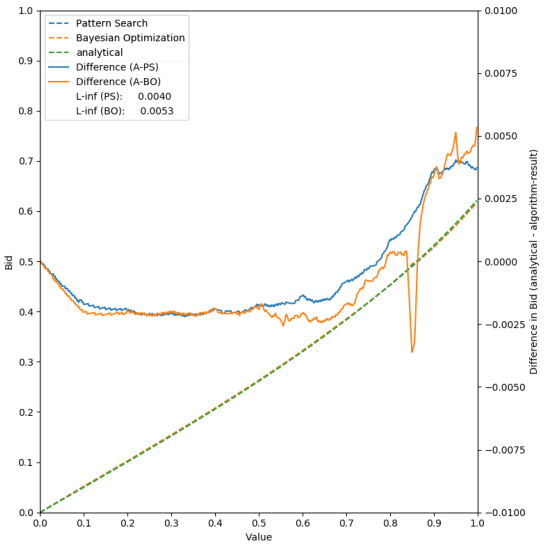
A.3.2 Quality of final Strategies

We compare the final strategies produced by the CA-BNE using PS and BO with the analytical results by [Ausubel and Baranov, 2018]. These are combined plots with two y-axis. The dotted lines belong to the left y-axis and represent the final equilibrium strategies for PS, BO and the analytical solution. The solid lines belong to the y-axis on the right and represent the difference between the found strategies (BO or PS) and the analytical solution. To be precise these graphs display analytical result minus found result.

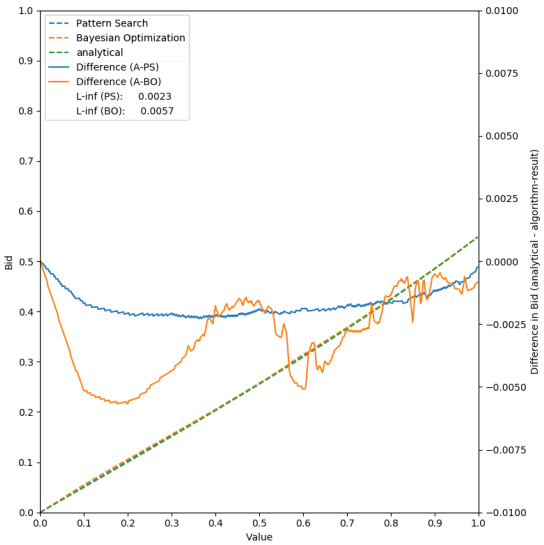


$\alpha = 1, \gamma = 0$

$\alpha = 1, \gamma = 0.5$

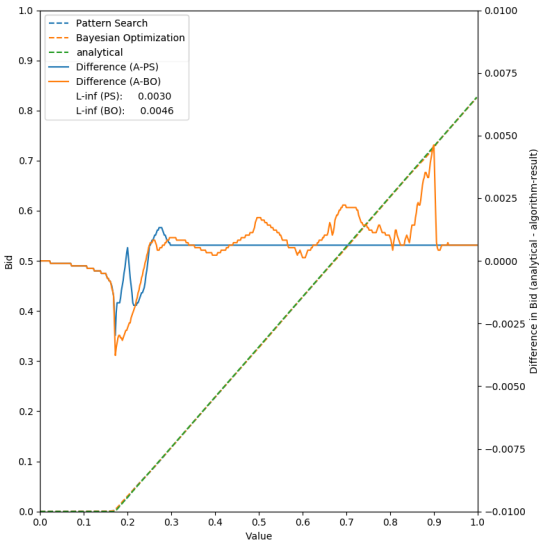


$\alpha = 2, \gamma = 0$

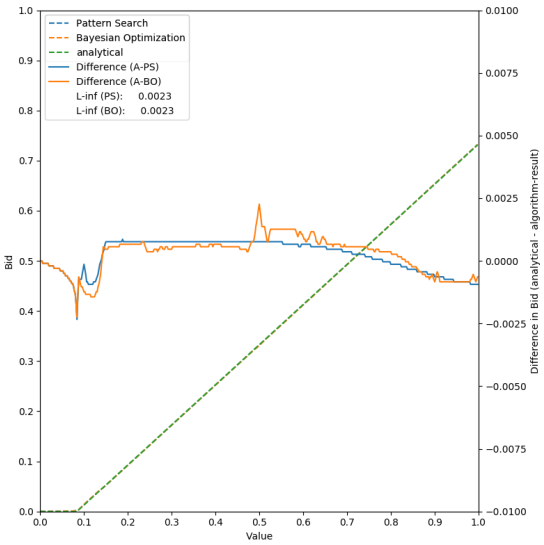


$\alpha = 2, \gamma = 0.5$

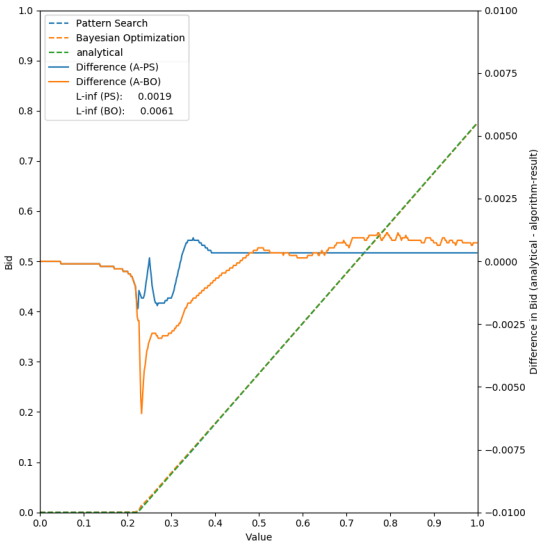
Figure A.33: **Comparison of found and analytical strategies for Nearest-Bid**



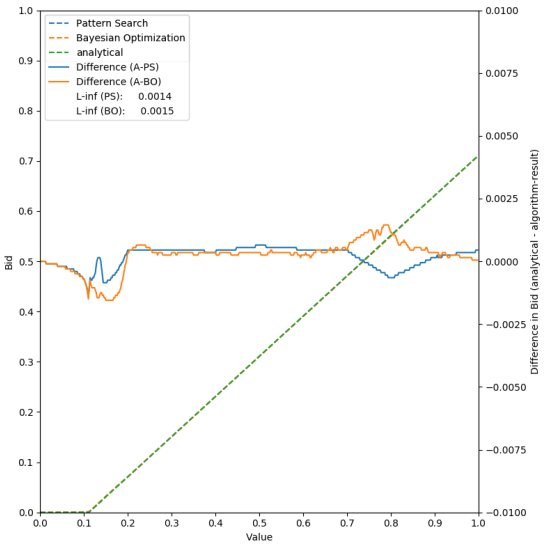
$\alpha = 1, \gamma = 0$



$\alpha = 1, \gamma = 0.5$

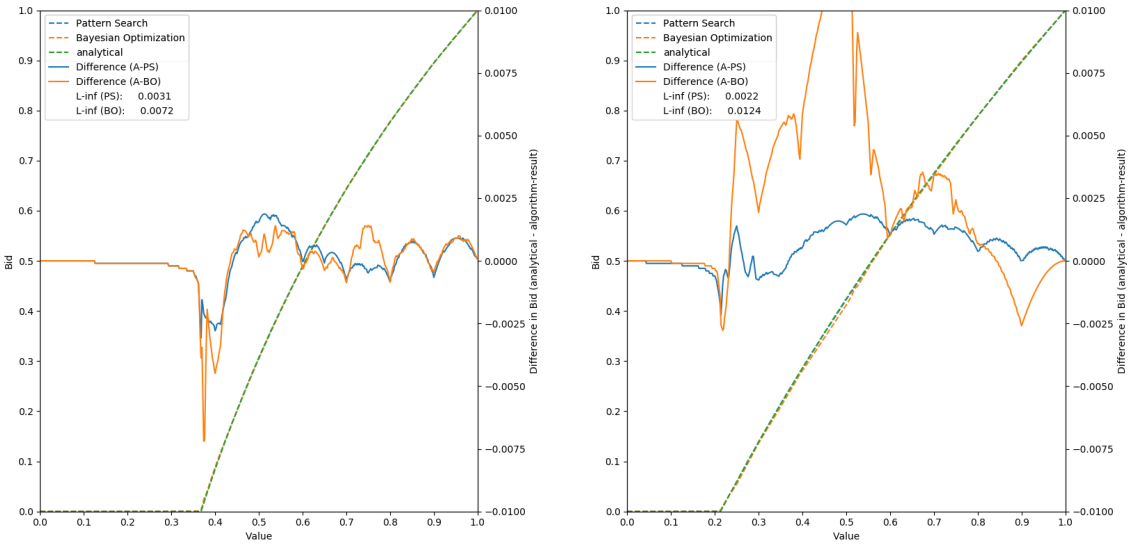


$\alpha = 2, \gamma = 0$



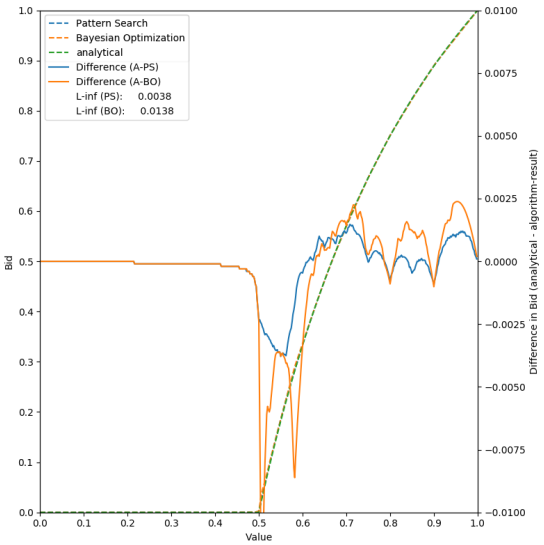
$\alpha = 2, \gamma = 0.5$

Figure A.34: **Comparison of found and analytical strategies for Proportional**

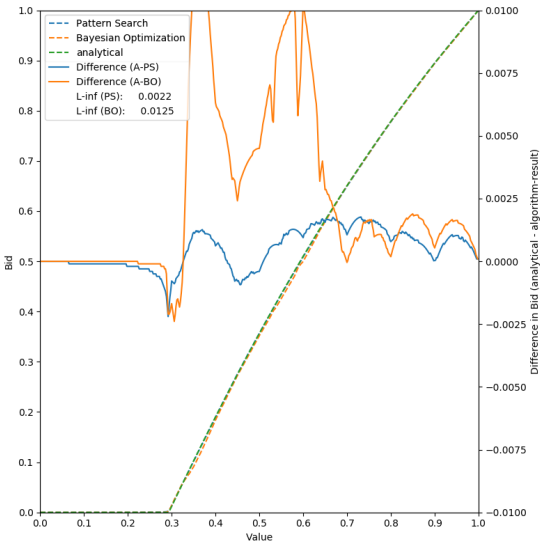


$\alpha = 1, \gamma = 0$

$\alpha = 1, \gamma = 0.5$

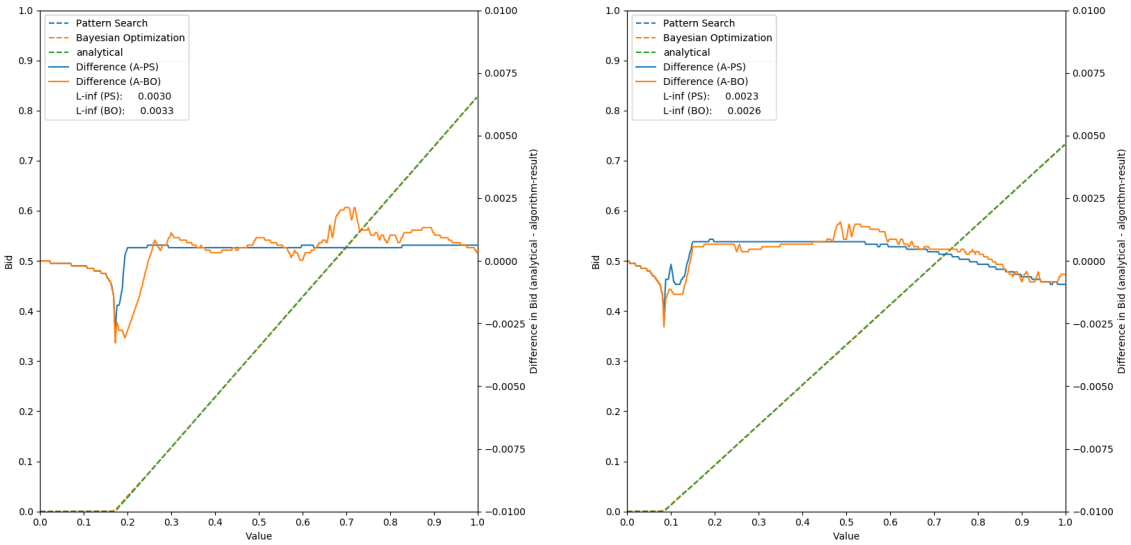


$\alpha = 2, \gamma = 0$



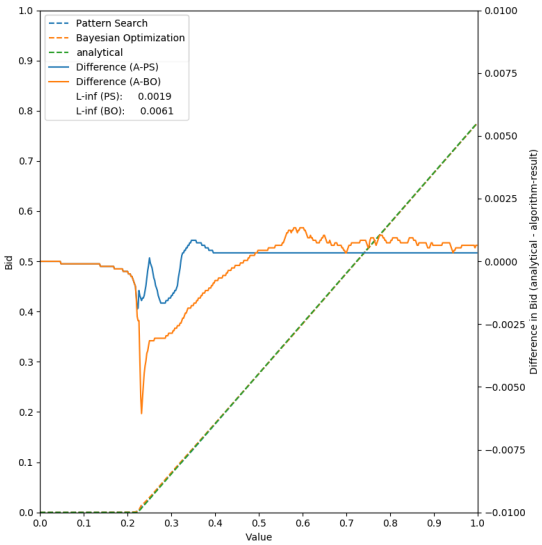
$\alpha = 2, \gamma = 0.5$

Figure A.35: **Comparison of found and analytical strategies for Proxy**

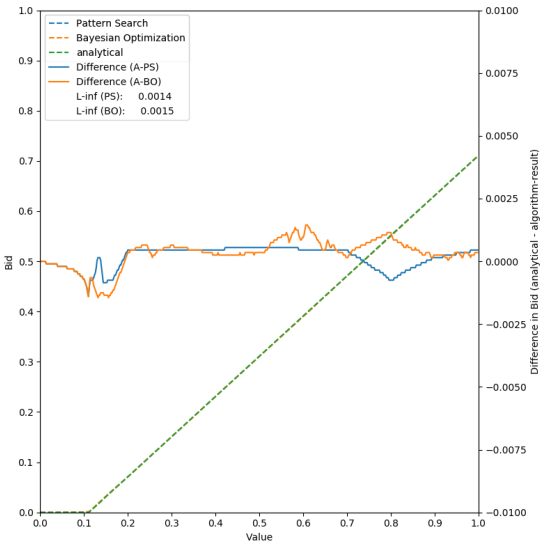


$\alpha = 1, \gamma = 0$

$\alpha = 1, \gamma = 0.5$



$\alpha = 2, \gamma = 0$



$\alpha = 2, \gamma = 0.5$

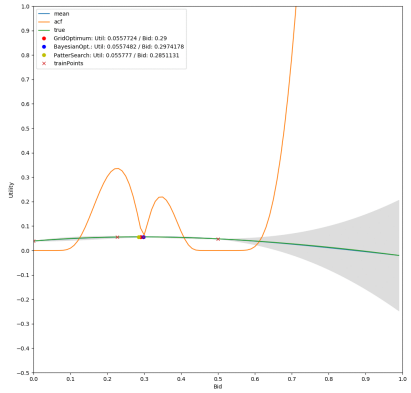
Figure A.36: **Comparison of found and analytical strategies for Quadratic**

A.3.3 BO Illustration

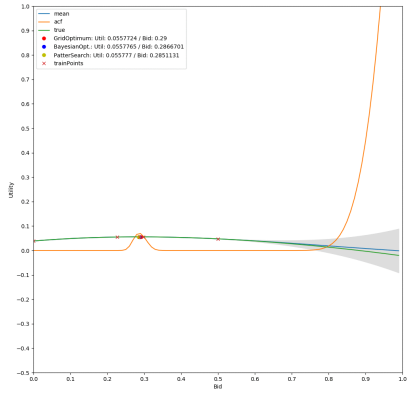
The subsequent plot are meant to give a better understanding how the BO works and which points are evaluated during the process. Displayed are two sequences of the first six iterations of the BO for the same domain. One time we use the Brent search for the acquisition function optimizing, the other time we use a grid search. For these plot the acquisition-value is scaled by 1000. Due to the enormous number of plots only a very limited selection is included in the printed version of this thesis.[3]

[3]The full set of plots can be found in on the CD that is handed in together with this printed version of the thesis.

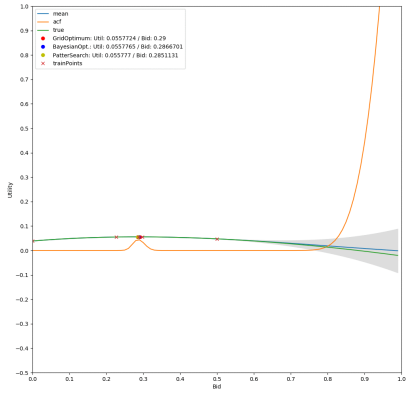
Nearest Bid $\alpha = 1, \gamma = 0$, BNE-iteration 2



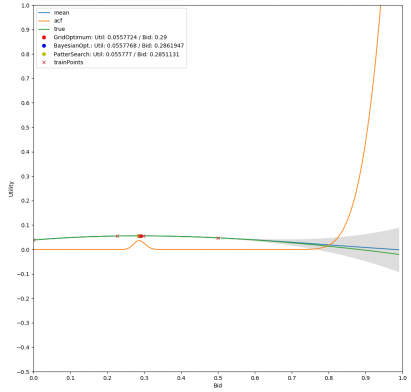
iteration 0



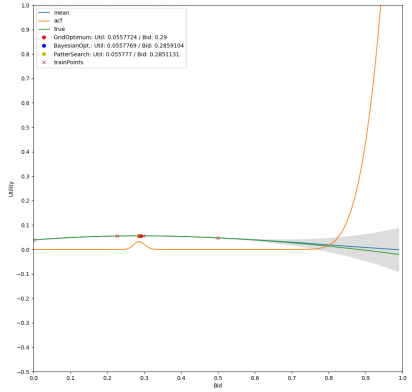
iteration 1



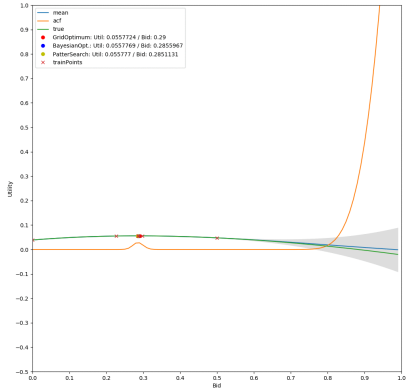
iteration 2



iteration 3



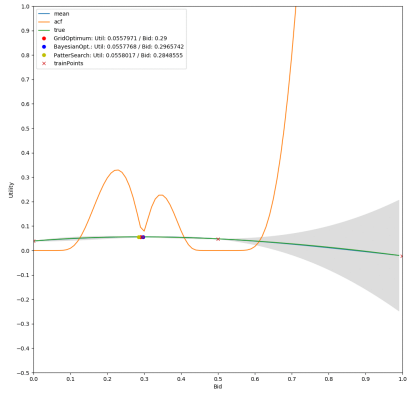
iteration 4



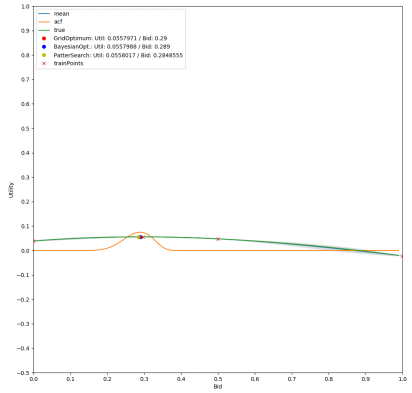
iteration 5

Figure A.37: **First 6 BO-iterations for Quadratic $\alpha = 1$, $\gamma = 0$**

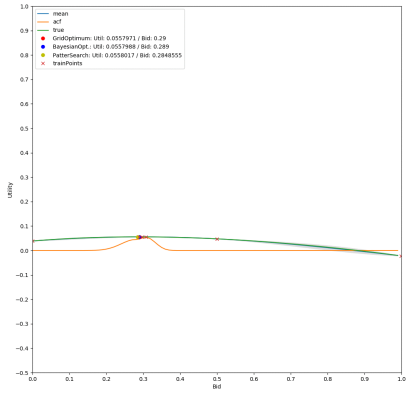
Nearest Bid $\alpha = 1, \gamma = 0$, BNE-iteration 2



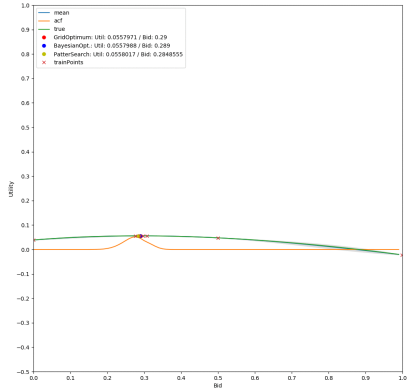
iteration 0



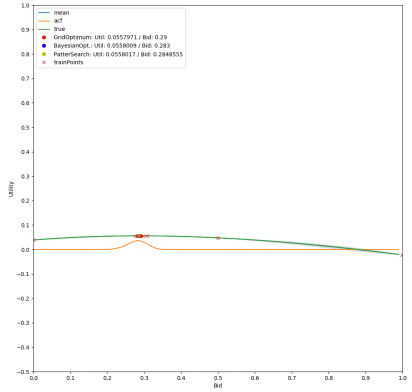
iteration 1



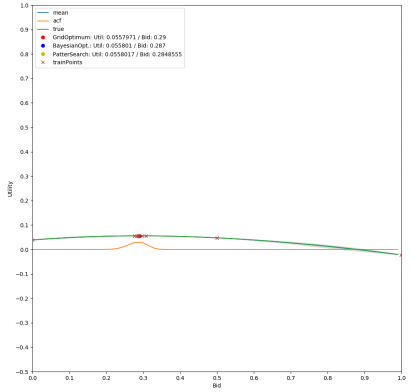
iteration 2



iteration 3



iteration 4



iteration 5

Figure A.38: **First 6 BO-iterations for Quadratic $\alpha = 1$, $\gamma = 0$, grid instead of Brent search**

List of Figures

List of Tables