Bazaar-Extension: A CloudSim Extension for Simulating Negotiation based Resource Allocations

Benedikt Pittl
University of Vienna
Faculty of Computer Science, Austria
benedikt.pittl@univie.ac.at

Werner Mach
University of Vienna
Faculty of Computer Science, Austria
werner.mach@univie.ac.at

Erich Schikuta
University of Vienna
Faculty of Computer Science, Austria
erich.schikuta@univie.ac.at

***Abstract*—Negotiation processes taking place between two or more parties need to agree on a viable execution mechanism. Auctioning protocols have proven useful as electronic negotiation mechanisms in the past. Auctions are a way of determining the price of a resource in a dynamic way. Additionally, auctions have well defined rules such as winner and loser determination, time restrictions or minimum price increment. These restrictions are necessary to ensure fair and transparent resource allocation. However, these rules are limiting flexibility of consumers and providers. In this paper we introduce a novel negotiation based resource allocation mechanisms using the offer-counteroffer negotiation protocol paradigm. This allocation mechanism shows similarities to the supermarket approach as consumer and provider are able to communicate directly. Further, the price is determined in a dynamic way similar to auctioning. We developed a *Bazaar-Extension* for CloudSim which simulates negotiation processes with different strategies. In this paper we introduce and analyze a specific Genetic Algorithm based negotiation strategy. For the comparison of the efficiency of resource allocations a novel *Bazaar-Score* was used.**

***Keywords*-SLA, Negotiation, Cloud Market, Bazaar**

I. Introduction

The importance of virtual value chains in enterprises increased dramatically. Virtual value chains as defined by [1] use information to create value, where services are utilized within the value adding steps of value chains. A digital market is the culmination point of stakeholders providing and requiring services used along each link in a value chain. Cloud computing by its characteristic of metered services described by negotiable Service Level Agreements (SLAs) paves the way to the realization of these digital markets [2]. In this paper we consider IaaS (Infrastructure as a Service) as an example of a cloud service. Thus we use the terms IaaS provider and cloud provider synonymously. Usually cloud providers run datacenters providing IaaS resources. Providers sell these resources to enterprises in form of virtual machines (VMs). Today, these machines are mainly traded directly for fixed prices from provider to consumer [3]. We use the term resource for all resources of a provider which can be offered in form of virtual machines. Virtual machines are not commodities which are totally interchangeable. A virtual machine is characterised using the following descriptors: (i) processing power (ii) storage (iii) RAM (iv) price. The processing power is provided by processing units and measured in million instructions per second (MIPS). RAM as well as storage are measured in Mega Bytes (MB).

In this paper we use the term resource allocation for a mechanism determining how providers and consumers sell and buy resources. One of the simplest allocation mechanism is the *supermarket* resource allocation mechanism [4]: Providers create offers which can be bought from consumers. Usually, the offers are not customized. Every consumer buys the same resource for a fixed price. A consumer selects the best fitting offer from the providers. The authors of [3] describe that a cloud provider usually will not be able to sell all its resources via such a fixed pricing mechanism. It seems like providers have to split their resources into two pools: one pool containing all resources sold via a fixed price mechanism and another pool containing the unsold resources. It is obvious that providers have to sell the unsold resource in order to optimize their profit. Resources which are not sold by a fixed pricing mechanism are called residual resources [3]. They can be sold using a dynamic pricing mechanism. Dynamic pricing mechanisms for residual resources are already business reality as Amazon's spot market [5] shows: consumers can bid for instances. The higher the bid, the higher is the chance of winning an instance.

For our research project focusing on economical aspects of Cloud Computing we needed an adaptive cloud simulation environment which is able to run resource allocation scenarios. We found the scientific simulator CloudSim and two relevant extensions using CloudSim for resource allocation simulation: (i) The extension introduced by [6] allows to run a double combinatorial auction. (ii) In [3] a CloudSim extension was introduced for executing procurement auctions.

The two approaches are auction based. To the best of the authors knowledge no extension exists enabling negotiation simulation and development of negotiation strategies. Therefore, we developed the *Bazaar-Extension* for CloudSim. A lot of papers describe interesting dynamic pricing resource allocation approaches but most of them are abstract and

978-1-5090-2628-9/16 $31.00 © 2016 IEEE
DOI 10.1109/SCC.2016.62

unimplemented in CloudSim. Implementation of different resource allocation mechanism fosters comparability. We see this paper as initial step to make different approaches comparable. In our future work we want to compare existing approaches using the Bazaar-Extension.

In [7] we introduced an approach for evaluating the utility of offers exchanged during negotiation which is used in the Bazaar-Extension. All utility functions used in this paper are based on [7]. The main contributions of this paper are: (i) description of the **Bazaar-Extension** architecture (section II) (ii) introduction of possible negotiation outcomes using a **genetic algorithm** (section III) (iii) description of a **simulation scenario** of a monopoly (section V).

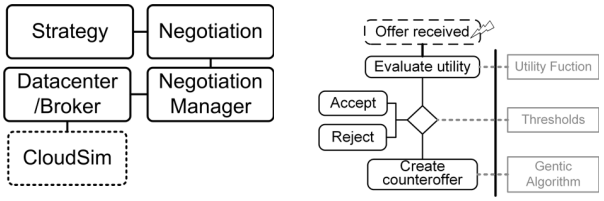
The architecture of the Bazaar-Extension is summarized in section II. The counteroffer generation using a genetic algorithm is described in the section III. In section IV the calculation of the market efficiency is described. An implemented negotiation scenario is introduced in section V. The used parameter setup for the scenario is described in the appendix [8]. The paper is closed by a description of further research in section VI.

II. Bazaar-Extension Architecture

The used negotiation style in the Bazaar-Extension is based on the offer and counteroffer mechanism described in the WS-Agreement Negotiation standard [4]. In a typical negotiation scenario two negotiation partners such as provider and consumer exchange messages (containing offers) in an alternating way: each negotiation partner sends offers, then it receives counteroffers to which it can respond again. This leads to a negotiation tree. The first message is usually called template. Templates are the initial offers published by the providers to promote negotiation [4]. Theoretically, a negotiation partner can create an arbitrary number of counteroffers in response to received offers. We use the term offer for both, a template as well as a counteroffer. An offer always contains a description of a VM. So we use these terms synonymously.The WS-Agreement Negotiation standard describes negotiation concepts but no concrete strategy. In the following sections a summary of the Bazaar-Extension as well as an initial negotiation strategy is introduced.

Using the Bazaar-Extension scientists are able to create a IaaS-market, add market participants to the market, assign negotiation strategies to them and analyse the resulting resource allocation using the Bazaar-Score.

The architecture of the Bazaar-Extension for CloudSim is shown in figure 2c. The Bazaar-Extension covers three main components. (i) The communication component contains the communication infrastructure so that negotiation messages can be exchanged between consumer and provider. (ii) Consumer and provider use negotiation strategies. In this paper a concrete strategy is presented using a genetic algorithm for counteroffer generation. (iii) The visualization component visualizes the negotiation results. The framework f(x)yz [9] was used for the 3D views.



(a) Simplified communication sequence

(b) Strategy overview

Figure 1: Overview over negotiation processes

Figure 1a depicts a high level view of the communication sequence of the Bazaar-Extension. The broker represents the consumer and the provider represents a datacenter so these terms are used synonymously. In CloudSim datacenter and broker do not communicate directly. Instead, they have to pass messages to a central component which is responsible for the delivery.

For the Bazaar-Extension a negotiation datacenter and a negotiation broker was created. For simplicity reasons we call a negotiation datacenter *datacenter* and a negotiation broker *broker*. The broker and the datacenter are so called entities. They extend the basic behaviour of the entity class defined in the CloudSim framework. Each datacenter and each broker has exactly one negotiation manager whereby each negotiation manager is responsible for several negotiations. A negotiation strategy is used for a negotiation.

Negotiation Manager: The negotiation manager is responsible for two tasks: (i) In CloudSim each entity has an address, called entity id, which is used for delivering messages. If a consumer needs a VM it can use usual CloudSim messages for sending negotiation offers to datacenters. By default, a CloudSim message contains the entity id of the sender so that receiver knows who was sending the offer.

If a consumer needs two VMs (for example a small and a large VM) its broker starts negotiations for each VM with a datacenter. Thus the broker has two independent negotiations with a datacenter at the same time. A receiver of a negotiation message is able to use the senders entity id for distinguishing between different entities. However, the receiver of negotiation messages is not able to distinguish between different negotiations with the same entity because messages do not contain a negotiation identifier. Therefore a bazaar negotiation message including a negotiation id was created. This bazaar negotiation message contains inter alia the offered VM and a negotiation id. Bazaar negotiation messages are encapsulated in usual CloudSim messages. For messages containing a Bazaar negotiation message a separate message tag was created which indicates the type of a message in CloudSim. One task of the negotiation manager is to forward messages to the corresponding negotiation. So the negotiation manager can be considered as a gate-

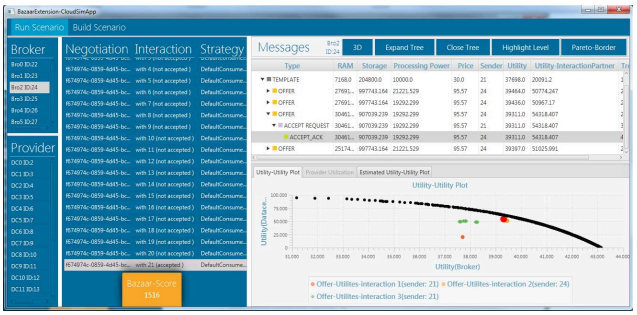
way. It checks if the received message contains a bazaar negotiation message using the message tag. In cases the received message encapsulates a bazaar negotiation message it reads the negotiation id to forward the message to the corresponding negotiation. This mechanism is similar to the IP/Port mechanism. The entity id is used like the IP address for sending messages to a destination. The negotiation id is similar to the port. It determines to which negotiation the message at the destination belongs. (ii) The second task of the negotiation manager is to create new negotiations if a negotiation message was received containing a negotiation id which is unknown by the negotiation manager.

Negotiation: The negotiation class is a container for storing negotiation relevant data. Negotiation relevant data is the negotiation id and a tree tracing the negotiation. Further a reference to the used strategy for the negotiation is stored.

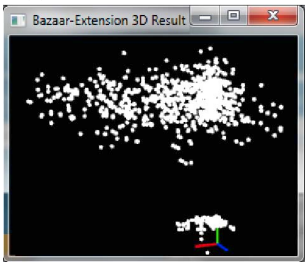
Negotiation Strategy: In this paper a concrete strategy for IaaS-markets is introduced. Figure 1b shows the basic structure of the initial negotiation strategy which is introduced in this paper. Each received offer is evaluated using utility functions which is done in the utility evaluator component shown in figure 2c. The result of the evaluation is a score created using the utility functions described in [7]. Based on this score thresholds decide if an accept (as described later in this paper there are two type of accept messages), reject or a counteroffer is created in response to the received offer. The thresholds are modified during negotiation based on market conditions (see section V) which is done in the threshold modifier component shown in figure 2c. The counteroffer is created using a genetic algorithm (see next section). The Counteroffer Generator component shown in figure 2c is responsible for this task.

It it is possible to develop other threshold modifiers, counteroffer generators or utility evaluators as described in this paper.Basically, any strategy can be developed using the Bazaar-Extension.

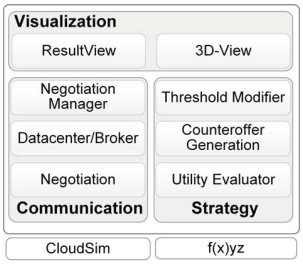
A screenshot of the Result-View of the Bazaar-Extension is depicted in figure 2a. The left side of the windows shows the consumers and providers attending the market scenario. By selecting a participant its negotiations including the used negotiation strategies are shown. In the Result-View the offers exchanged during a negotiation are represented as tree list and visualized in a so called utility-utility plot (see right side of the window). Consumer and provider evaluate and rank received offers by assigning utility values to them. The axis of the utility-utility plot represent the utility value of an offer for consumer and provider. For assigning utility values to offers utility functions as described in [7] were used which consider economical principles from consumer theory as e.g. described in [10]. By using utility functions a Pareto-boarder can be calculated which is shown in the utility-utility plot in figure 2a. The orange box in figure 2a contains the Bazaar-Score of the executed scenario. Negotiation results are visualized in a 3D-plot as shown in figure 2b. The red, green and blue axis represent the processing power, RAM and storage.



(a) Result-View of CloudSim Bazaar-Extension - Screenshot



(b) 3D Result-View of Bazaar-Extension - Screenshot



(c) High level architectural view of the Bazaar-Extension

Figure 2: Architecture and Result-View of Bazaar-Extension

III. Creating Counteroffers

An agreement is formed if a counteroffer has a high utility for the sender and for the receiver. The creation of such counteroffers is a quite complex task. This is because the utility of counteroffers has to be optimized according to given preferences reflecting the overall business strategy of a negotiator. As a comprehensive business strategy considers elements like risk assessment, customer relationship and business rules (e.g. compliance rules) an analytical based optimization will be difficult and time intensive. Due to the algorithmic complexity of calculating the utility of all possible counteroffers we propose a heuristic approach fostering genetic algorithms. In this section we want to emphasize the counteroffer creation based on a genetic algorithm by representing business strategies by simplified utility functions. We also considered the usage of other meta heuristics and knowledge engineering techniques like neural networks. However, genetic algorithms seem to be well suited for our problem because they (i) need no training phase (ii) can be easily parametrized using the fitness function and (iii) creates counteroffers as byproduct.

The authors of [11] and [12] use a genetic algorithm where the population consists of a set of strategies whereby each strategy consists of a set of predefined rules. For the highly dynamic service markets dynamic strategies seem to be more successful than strategies with predefined rules. The

authors of [13] use a genetic algorithm for assigning sellers to buyers. The allocation is optimal if buyer and same have the same valuation of a good described by attribute i. A so called match value (v_i) for an attribute i is calculated using the equation $v_i = 1 - |b_i - s_i|$ whereby b_i represents the value of attribute i for the buyer and s_i represents the value for the seller. As this equation shows the greater the difference of the valuation of buyer and seller the lower is the match value. The final fitness value is formed by weighting all match values.

For our Bazaar-Extension we are not using this approach because of the following reasons: (i) We assume that a consumer does not know the providers valuation of an attribute and vice versa. (ii) We think that utility functions are eligible to present consumer and provider valuation/preferences. Utility values resulting from different utility functions are not comparable. Thus the difference of the consumer and provider valuation can not be calculated as shown in equation before (utility values are ordinal scaled).

We have not found a genetic algorithm for creating offers. Hence, we developed our own genetic algorithm using the idea of considering the valuation of negotiation partners in the fitness function as introduced in [13]. By assuming that the utility functions of the negotiation partners are unknown the creator of a counteroffer has to estimate the utility functions used by the negotiation partners. The genetic algorithm introduced in this section is used by the Bazaar-Extension during negotiation simulation.

We describe VM characteristics using a vector like (x_1, x_2, x_3, x_4). The first element (x_1) represents the storage (GB), the second element the processing power (MIPS), the third element RAM (GB) and the last element the price ($). A genetic algorithm has a Population, a Fitness Function as well as Crossover and Mutation operations.

Population: The population of the genetic algorithm consists of vectors representing VMs. These VMs are the individuals. There are two basic options for creating the initial population.

1) The received offer is ignored for population generation. So the initial population is created randomly.
2) Generally a received offer has high utility for its sender. Hence the received offer is the basis for creating the population and thus a counteroffer.

Individuals resulting from a random created initial population using option 1) may have no utility for the negotiation partner as described in the next paragraph. Hence we decided to create the population by using option 2) based on the received counteroffer: Depending on the population size, different variations of the received offer are created which form the initial population. An individual is created by modifying one of the four characteristics of a VM. For example, an individual differs in price from received offer whereas another individual differs in storage from the received offer. Characteristics are increased as well as decreased. Due to crossover and mutation operations the offers created by the genetic algorithm have less similarity with the received offer. But the result is more similar to the received offer than it would be by using a random initial population.

Fitness Function: The used fitness functions for counteroffer creation have a twofold goal: they have to represent the utility for the sender as well as the receiver. So the used fitness function has two components. The first component represents the utility function used by the sender and the second component is an estimation function representing the utility for the negotiation partner. The estimated fitness function may be generated using genetic programming techniques. Techniques for creating estimated utility functions are part of our further research. All estimated utility functions $\bar{U}$ were defined by us to illustrate the mechanism of the negotiation strategy. Individuals having a high fitness value usually have a high utility for both consumer and provider.

It has to be noted that the estimated utility function representing the utility of the negotiation partner is imprecise. So, an offer with a high fitness value may be not acceptable for the negotiation partner because the high fitness value may result from the imprecise estimated fitness function. Therefore, we limited the initial population creation by using option 2) to keep the counteroffer closer to the received offer. The received offer is used as guideline for counteroffer creation. This reduces the risk of creating offers with high fitness values and low utility for the receiver.

The basic structure of the used fitness functions are shown in equation (1). The fitness function $F_{consumer}$ is used by consumer. It considers its utility function as well as an estimated utility function of the provider $\bar{U}_{provider}$. Similar the fitness function used by the provider considers its utility function and an estimated utility function of the consumer $\bar{U}_{consumer}$. The estimated utility functions have to be weighted with weight w for an adequate share. The higher the weight, the stronger is the consideration of the negotiation partner. Therefore w is called *consideration factor*. For the scenarios we have defined the size of w.

$$\begin{aligned} F_{consumer} &= U_{consumer} + \bar{U}_{provider} \cdot w \\ F_{provider} &= U_{provider} + \bar{U}_{consumer} \cdot w \end{aligned} \tag{1}$$

Crossover and Mutation: The creation of a new generation encompasses two basic steps.

(i) Elitism is used for creating a part of the new generation by putting the best individuals regarding fitness of the old generation to the new generation. The selected individuals are not modified any more. (ii) The other individuals are generated using crossover and mutation operations. A Roulette Wheel Selection is used for parents selection needed during the crossover operation. Thus, the parent selection probability is proportional to the fitness of an individual:

$$P_i = \frac{F_i}{\sum_{n=0}^{p} F_n} \tag{2}$$

Table I: Genetic algorithm setup summary

Parameters	Values	Parameters	Values
Population Size	96	Mutation Probability	5%
Received VM	(200,10000,7,30)	Elitism	best 5%
Fitness Function			
$U_x = \begin{cases} log(x) & x \geq Min_x \\ -\infty, & x < Min_x \end{cases}$		x $\in \{RAM, Storage, Proc.Power\}$	
$U_{Price} = \begin{cases} log(Max_{Price} - Price + 1) & Price \leq Max_{Price} \\ -\infty, & Price > Max_{Price} \end{cases}$			
U=1 $\cdot U_{Price} + 1 \cdot U_{RAM} + 1 \cdot U_{Storage} + 1 \cdot U_{Proc.Power} + 100000$			
$\bar{U} = Price - RAM \cdot 0.001 - Storage \cdot 0.0005 - Proc.Power \cdot 0.001$			
$w = 25$			

P_i is the selection probability of an individual i, F_i is the fitness of the individual i and p is the population size.

After the selection of two parents an individual is created by taking randomly two characteristics of the first parent and the other characteristics of the other parent. The new generated individual may be mutated by modifying one of its characteristic.

The best offers created by the algorithm are sent as counteroffers. The developed genetic algorithm was evaluated by running several testes. Some of them are introduced in the following paragraph. Table I shows the used setup parameters. The utility functions U and $\bar{U}$ are based on economical principles described in [7]. In this paper the structure of the utility functions is described in more detail. The utility function $\bar{U}$ represents an estimation of a typical utility function used by a provider which is representing the profit contribution. For example 0.001 is the price for one MB RAM. U is a typical example of a utility function used by a consumer which increases with additional resources. A small iteration and population size was used for the tests as a quick counteroffer generation is necessary during negotiation. For all tests it was assumed that the offer (200, 10000, 7, 30) is received. Consequently this offer was the basis for population generation.

Table II contains eight parameter setups $S1 - S8$ used for running the genetic algorithm. In four setups 50 generations (iterations) were created whereas in the other four setups 100 generations were generated. The genetic algorithm was evaluated using the average fitness value as suggested by [14]. Therefore, the genetic algorithm was executed with each setup 1000 times. After each execution the fitness of the best individual was stored. The average fitness value of best individuals is shown in the table II. The performance of the setups are visualized in figure 3. The first line represents the fitness value of the received offer. The initial offer has a fitness of 132338.8. The boxplots represent fitness values of the setups by executing each algorithm with each setup 1000 times. The median of the best fitness values of all experiments exceeds the fitness value of the received offer. Elitism and mutation was not use in the setups $S4$ and $S8$. Sometimes the best individual returned by executing the

Table II: Genetic algorithm scenarios and results

	Generations	
	50	100
with Mutation/Elitism	*S1*-134453.7	*S5*-134669.3
without Mutation with Elitism	*S2*-134134.7	*S6*-134142.9
with Mutation without Elitism	*S3*-133058.9	*S7*-133006.6
without Mutation/Elitism	*S4*-132831.4	*S8*-132654.6

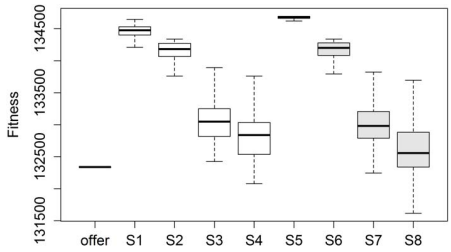
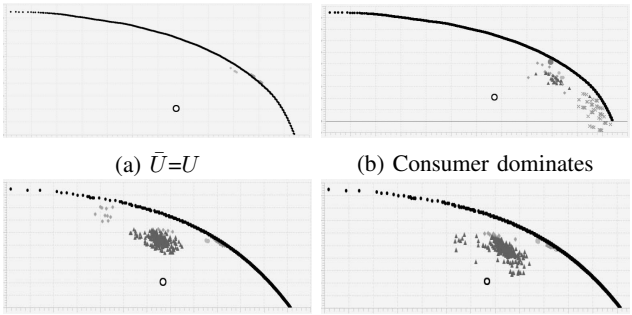


Figure 3: Genetic algorithm scenarios

algorithm using one of these setups has a lower fitness than the received offer. The usage of Mutation as well as Elitism seems to improve the result. $S5$ is the best setup according to this evaluation. For the negotiation scenario presented in section V we used the setup of $S5$ for counteroffer generation.

In figure 4 some negotiation examples using different consideration factors are shown. In all graphs the ordinate represents the utility of the datacenter and the abscissa represents the utility of the broker. The template is represented by a white point with a black boarder and the message exchanged between consumer and provider are visualized by grey points. In all figures the black points forming an boarder represent an approximate Pareto-boarder. Messages on that boarder are Pareto-optimal. In figure 4c the consumer uses a high consideration factor while in figure 4b the provider uses a high consideration factor. In the negotiations 4a and 4d consumer as well as provider use a moderate consideration factor. Consumer and provider use real utility functions ($\bar{U} = U$) in the negotiation depicted in figure 4a. So nearly



(a) $\bar{U}$=U (b) Consumer dominates

(c) Providers dominates (d) Balanced negotiation

Figure 4: Screenshot of the Bazaar-Extension Result-View

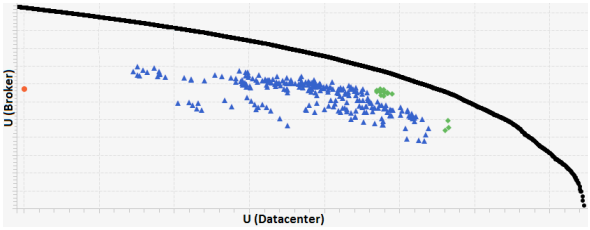


Figure 5: Utility-utility plot - global perspective

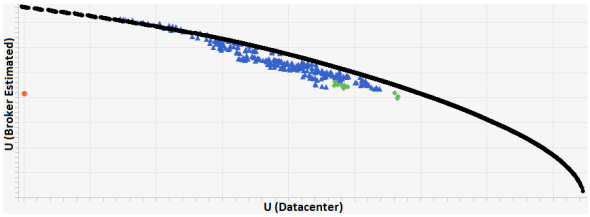


Figure 6: Utility-utility plot - provider perspective

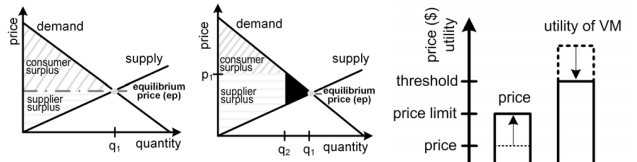
all points are on the Pareto-boarder. Estimated functions ($\bar{U} \neq U$) are used in the negotiation visualized in figure 4d.

A typical negotiation is shown in figure 5. The ordinate represents the utility of the broker while the abscissa represents the utility of the datacenter. The points in the plot show all messages created by the provider. The red dot represents the template. Consumers create counteroffers in response to the template to which the provider responses with messages represented by green points. After the counteroffers are received, the provider responds to them with counteroffers visualized as blue triangles. A lot of offers have a great distance to the Pareto-boarder. The distance occurs because (i) the genetic algorithm calculates approximations and (ii) the estimated utility function $\bar{U}$ is imprecise.

Figure 6 visualizes the perspective of the datacenter where the ordinate represents the estimated utility ($\bar{U}_{consumer}$) of the consumer. The offers created by the provider are approximately Pareto-optimal from the perspective of the provider.

IV. Measuring Efficiency of Markets

The goal of the Bazaar-Extension is to simulate and analyse different negotiation strategies and resource allocations. Therefore a key figure measuring the efficiency of different resource allocations is needed. Based on a literature survey focusing on how to measure efficiency in markets we found the approach introduced by [10]. The authors of [10] assume a typical market model starting from the premise that all consumers and buyers are trading identical goods. The demand curve represents the market demand for that good at a certain price. Consumers buying a good at the current price are called *buying consumers*. Buying consumers buy a good because, for them, the value of the good exceeds the



(a) Valuation of a good (b) Surplus after price increment (c) Price limit calculation

Figure 7: Market surplus and efficiency

price. C is a set containing all consumers. v_i is the value of the good for a consumer $C_i \in C$. The set of buying consumers can be expressed by the following equation:

$$C^{\text{buying consumers}} = \{C_i \in C | v_i > price\} \quad (3)$$

The equilibrium price ep is the price at which demand matches supply. Usually the market efficiency is maximised at the equilibrium price. The quantity sold at the equilibrium price is marked with q_1 in figure 7a. Further consumers are willing to buy a good. However, for them the value of the good is lower than the price. Consequently no further sale transaction happens. The demand curve represents the value of consumers of a good [10]. The difference between value and price is called consumer surplus and is represented by the upper part of the grey area in figure 7a. The leftmost part of the demand curve represents consumers valuing the good very high, i.e. they have a great consumer surplus at the current price eq. The marginal consumer is a consumer, who assesses the good at the current price. If the price increases the marginal consumer will not buy the good any more. The total consumers surplus is the sum of all consumers surpluses as shown in the following equation:

$$\text{total consumer surplus} = \sum v_i - price, C_i \in C^{\text{buying consumers}} \quad (4)$$

A change of a price results into a change of total consumer surplus. For example if the current price p_{old} decreased to p_{new} the consumers surplus increases because two effects show up. New consumers $C_i^{new} \in C | v_i < p_{old} \wedge v_i > p_{new}$ will buy the good so that the total quantity sold increases. These new consumers buy the good because the reduced price is lower than their value of the good leading to additional sales and consequently to an additional consumer surplus. (i) The buying consumers who would buy the good for the old price $C_i^{old} \in C | v_i > p_{old}$ get a greater surplus as their surplus increases from $v_i - p_{old}$ to $v_i - p_{new}$. The surplus difference Δ depends on the price difference $\Delta = p_{old} - p_{new}$.

The supply curve represents the cost of suppliers assuming diminishing marginal product [10]. Similar to consumers, suppliers sell a good if their cost is lower than the price as depicted in figure 7a. The difference between cost and price is called supplier surplus. The sum of the total consumer surplus and the total supplier surplus is named total surplus

and is used by [10] measuring market efficiency. Consider the example shown in figure 7b, where the price is set to p_1 which is above the equilibrium price. This price increment has the following two effects. (i) The quantity sold decreases to q_2. This is because some consumers do not buy the product any more. The set of remaining buying consumers can be defined as shown in the following equation:

$$C_{remaining}^{\text{buying consumers}} = C_i \in C | v_i > p_1 \tag{5}$$

All remaining buying consumers lose some consumers' surplus due to the price increment. At the same time suppliers gain supplier surplus. For the remaining buying consumers and suppliers the market efficiency measured by the total surplus is not affected. The increasing supplier surplus compensates the reduced consumer surplus of the remaining consumers. (ii) Some consumer are not buying the product any more due to the price increment. The set of consumers not buying the product any more is defined in the following equation:

$$C_{lost}^{\text{buying consumers}} = C_i \in C | v_i < p_1 \wedge v_i > \text{ep} \tag{6}$$

None of the consumers $C_i \in C_{lost}^{\text{buying consumers}}$ has a surplus any more as they stop buying. Consequently suppliers are not able to sell as much goods as in the previous scenario where the price was equal to ep. The lost surplus resulting form the non occurring sale transactions is highlighted by a black triangle in figure 7b. Due to the decreased number of sales transactions represented by this triangle the total surplus is smaller than in the previous scenario where the price was equal to the equilibrium price.

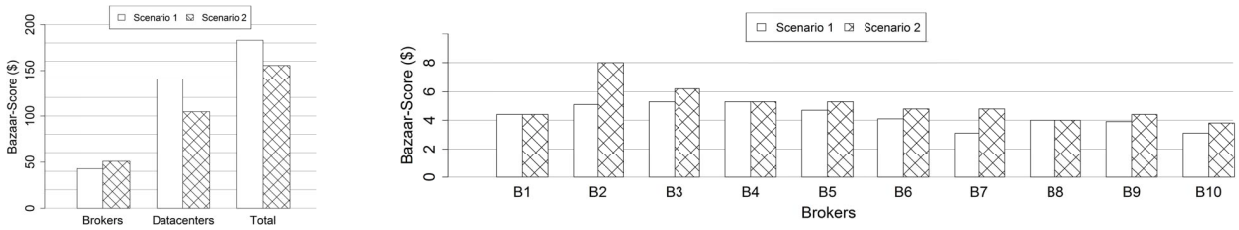
Our initial idea was to use utility functions for comparing market efficiency. Utility functions return utility values which are ordinal scaled and so utility values resulting form different utility functions are not comparable. So they are not adequate for our purpose. The market efficiency described in [10] is measured using the price. Thereby the authors assume a market model where only one good is traded for which all participants pay/charge a single price. VMs are no commodities as they have several characteristics. The characteristics of a VM determine the price consumers will pay and the cost of a provider hosting such a VM. Thus, the market model from [10] cannot be directly applied to IaaS-markets. Our goal is to make different resource allocations comparable and so we defined the *Bazaar-Score*. Similar to the surplus described above there is a consumer as well as a provider Bazaar-Score. The Bazaar-Score measures the difference between the price paid and the value of a VM for the consumer/the cost of a VM for the provider. Consumers and providers use utility functions for evaluating received offers. If an offer has a high utility for its receiver, then the receiver tries to form an agreement. The utility limit at which a receiver tries to form an agreement is called utility acceptance threshold. If the utility value of an offer exceeds the utility acceptance threshold then the receiver tries to make an agreement. An agreement contains the characteristics of a VM including the price. Usually an increasing price reduces the consumers utility. So if the price for the VM described in the agreement is increased, the utility of the VM for the consumer is decreased. The price at which the utility of the agreement is equal to the utility acceptance threshold is called upper price limit, as shown in figure 7c. At this price limit the consumer will be undecided between buying and not buying the VM. The utility of the VM priced with the price limit is equal to the utility acceptance threshold.

The difference between the price stated in the agreement and the upper price limit at which the utility of the VM equals the utility acceptance threshold is the consumer Bazaar-Score.The negotiation strategy which we developed distinguishes inter alia between offers, accept requests and accept acknowledges. In order to form an agreement a negotiation partner has to send an accept request message in response to an received offer. The receiver of the accept request message can respond with an accept acknowledge to form an agreement or with counteroffers for continuing negotiation. During negotiation we use two utility acceptance thresholds. (i) **Accept request threshold.** If an offer is received which has an utility exceeding the accept request utility threshold then the receiver tries to from an agreement. So it sends an accept request to the negotiation partner. (ii) **Agreement threshold.** If an accept request is received then the receiver can directly form an agreement. An agreement is formed if an accept request is received which exceeds the agreement utility threshold. If an offer is received exceeding the agreement utility threshold then the consumer responds with usual counter offers. So the agreement threshold is only considered if an accept request message arrives. The risk of not finding an agreement can be avoided by accepting a lower utility. In our simulation we used a so called reject threshold. Offers having a lower utility than the reject threshold are rejected. In the scenarios described in the following sections we predefined the values of the thresholds. The thresholds may change over time depending on the current negotiation states of a consumer or provider.

Similar to the consumer Bazaar-Score the provider Bazaar-Score is calculated. The provider evaluates offers with its utility function. Generally, the utility of the offer increases for a provider with increasing price. So the provider Bazaar-Score is calculated by decreasing the price of the agreement until its agreement threshold is reached. The Bazaar-Score is calculated. The total surplus is calculated by summing consumer and provider Bazaar-Score.

V. Bazaar-Extension Simulation Scenarios

In this section two scenarios called Scenario 1 and Scenario 2 are presented using the genetic algorithm and the Bazaar-Score introduced in this paper. Both scenarios use



(a) Bazaar-Score of market participants

(b) Bazaar-Score of brokers

Figure 8: Scenario results

the same negotiation strategy with different parameters. The most important parameters are described in appendix [8]. Both scenarios encompass 10 brokers and 1 one datacenter. So a monopoly is simulated. Scenario 1 and Scenario 2 use a different w for $\bar{U}_{consumer}$, all the other parameters are identical. In Scenario 2 the datacenter is more cooperative than in Scenario 1. The minus points shown in the table are used by the providers. If no acceptable offer was received form a consumer then the accept request threshold $U^{threshold}_{\text{accept request}}$ is decreased by the minus points for this negotiation. Thus the chance of receiving an acceptable offer is increased for further offers send by the consumers and received by the provider.

Figure 8b shows the Bazaar-Score of the brokers in the two scenarios. Some brokers can improve their Bazaar-Score in Scenario 2, while other brokers have an identical Bazaar-Score in Scenario 1 and Scenario 2. This is because a seed was used for the random variable utilized in the genetic algorithm. A summary of the market outcome of the two scenarios is shown in figure 8a. In Scenario 1 where the datacenter is less cooperative the Bazaar-Score is higher than in Scenario 2. However the brokers are better off in Scenario 2 than in Scenario 1. Regarding the Bazaar-Score Scenario 1 is more efficient than Scenario 2. So Bazaar-Score gain of the providers in Scenario 2 can not compensate the providers Bazaar-Score loss. A different resource allocation may occur by modifying the setup parameters. So different economical concepts as for example price elasticity such as described in [10] can be simulated.

VI. Conclusion and Further Research

This paper introduces the Bazaar-Extension which allows to simulate markets based on bilateral negotiations. Bilateral negotiations help to form highly customized SLAs. The negotiations results were analysed using the Bazaar-Score. A critical issue for automatic negotiation is the parameter setup used for the negotiation strategy. In our further research we want to use historical negotiation results for finding an adequate parameter setup. Further we will work on approaches helping to identify the estimated utility function.

References

[1] J. F. Rayport and J. J. Sviokla, "Exploiting the virtual value chain," *Harvard business review*, vol. 73, no. 6, p. 75, 1995.

[2] I. U. Haq, E. Schikuta, I. Brandic, A. Paschke, and H. Boley, "Sla validation of service value chains," in *Grid and Cloud Computing, International Conference on*. IEEE, 2010, pp. 308–313.

[3] P. Bonacquisto, G. D. Modica, G. Petralia, and O. Tomarchio, "A Strategy to Optimize Resource Allocation in Auction-Based Cloud Markets," in *Services Computing (SCC), 2014 IEEE International Conference on*. IEEE, 2014, pp. 339–346.

[4] O. Waeldrich, D. Battr, F. Brazier, K. Clark, M. Oey, A. Papaspyrou, P. Wieder, and W. Ziegler, "Ws-agreement negotiation version 1.0," in *Open Grid Forum*, 2011.

[5] "AWS | Amazon Elastic Compute Cloud (EC2) Cloud Server." [Online]. Available: //aws.amazon.com/de/ec2/

[6] P. Samimi, Y. Teimouri, and M. Mukhtar, "A combinatorial double auction resource allocation model in cloud computing," *Information Sciences*, 2014.

[7] B. Pittl, W. Mach, and E. Schikuta, "A negotiation-based resource allocation model in iaas-markets," in *Utility and Cloud Computing (UCC) (accepted on 06.09.2015)*, 2015.

[8] "Appendix." [Online]. Available: http://homepage.univie.ac.at/a1347629/appendix.pdf

[9] "F(X)yz by Birdasaur." [Online]. Available: http://birdasaur.github.io/FXyz/

[10] N. Mankiw, *Principles of Economics*. Cengage Learning, Jan. 2014.

[11] J. R. Oliver, "On artificial agents for negotiation in electronic commerce," in *System Sciences, 1996., Proceedings of the Twenty-Ninth Hawaii International Conference on,*, vol. 4. IEEE, 1996, pp. 337–346.

[12] M. T. Tu, E. Wolff, and W. Lamersdorf, "Genetic algorithms for automated negotiations: A fsm-based application approach," in *Database and Expert Systems Applications, 2000. Proceedings. 11th International Workshop on*. IEEE, 2000, pp. 1029–1033.

[13] S. A. Ludwig and T. Schoene, "Matchmaking in multi-attribute auctions using a genetic algorithm and a particle swarm approach," in *New Trends in Agent-Based Complex Automated Negotiations*. Springer, 2012, pp. 81–98.

[14] K. Sugihara, "Measures for Performance Evaluation of Genetic Algorithms (Extended Abstract)," in *Proc. 3rd Joint Conference on Information Sciences (JCIS '97*, 1997, pp. 172–175.