2013

Energy-aware Service Allocation for Cloud Computing

Tobias Widmer
University of Hohenheim, Department of Information Systems 2, Stuttgart, Germany, tobias.widmer@uni-hohenheim.de

Marc Premm
University of Hohenheim, Department of Information Systems 2, Stuttgart, Germany, marc.premm@uni-hohenheim.de

Paul Karaenke
University of Hohenheim, Department of Information Systems 2, Stuttgart, Germany; FZI Forschungszentrum Informatik, Karlsruhe, Germany, paul.karaenke@uni-hohenheim.de

Energy-aware Service Allocation for Cloud Computing

Tobias Widmer[1], Marc Premm[1], and Paul Karaenke[1,2]

[1] University of Hohenheim, Department of Information Systems 2, Stuttgart, Germany
{tobias.widmer,marc.premm,paul.karaenke}@uni-hohenheim.de
[2] FZI Forschungszentrum Informatik, Karlsruhe, Germany

Abstract. Energy efficiency has become an important managerial variable of IT management. Whereas cloud computing promises significantly higher levels of energy efficiency, it is still not known, if and to what extent outsourcing of software applications to cloud service providers affects the overall energy efficiency. This research is concerned with the allocation of cloud services from providers to customers and addresses the problem of energy-aware service allocation. The distributed nature of the problem, i.e., the multiple loci of control, entails the failure of centralised solutions. Hence, we approach this problem from a multiagent system perspective, which preserves the distributed setting of multiple service providers and customers. The contribution of our research is a game-theoretic framework for analysing service provider and customer interactions and a novel distributed allocation mechanism based on this framework to approximate energy-efficient, optimal allocations. We demonstrate the usefulness and efficacy of the proposed artifact in several simulation experiments.

Keywords: Game Theory, Green IT, Multiagent Systems, Negotiation, Resource Allocation

1 Introduction

Cloud-based service providers (SPs) such as Amazon and Google offer a portfolio of cloud services. Customers rent internet-accessible computing resources including CPU time and data storage services. Customers reserve such resources on demand without having to face particular operational setup costs [1] and are commonly charged according to the pricing-schemes “pay-per-use” or subscriptions [2]. Cloud computing, however, does not only promise significant cost reduction by lower IT system maintenance and operational costs, but also provides potential for increased energy efficiency, since the resources of large data centres can be managed more efficiently [3].

We consider a setting where customers intend to outsource software applications to cloud-based SPs to reduce costs and energy consumption of their IT systems. Therefore, we address the problem of *service allocation* from SPs to customers, i.e., the allocation of individual customer requests to matching available services. When allocating scarce resources, a typical objective is to find allocations which are optimal regarding a metric that depends on the preferences of the individual actors. Individual preferences can be aggregated using the notion of *social welfare* from welfare economics [4]. We assume that individual actors model their preferences by utility functions and thus apply the concept of *utilitarian social welfare*, denoting the sum of all individuals' utility for a given allocation. However, the problem of finding optimal allocations is often computationally infeasible in network settings such as in cloud computing with multiple SPs and customers, since computing optimal allocations is *NP*-complete [5].

Resource allocation is a well established field in multiagent systems research [6]. Thus, each actor (i.e., SPs and customers) is represented as a software agent in the proposed approach. These agents are able to perform autonomous actions in order to pursue its individual objectives [7]. Participating agents reason about the *processes of coordination* among themselves and negotiate bilateral agreements [8]. Existing approaches, however, do not consider energy efficiency, do not address utilitarian social welfare maximisation, or require a central coordinating agent which does not exist in the setting considered.

We develop a distributed heuristic for energy-aware cloud service allocation and evaluate this artifact in a set of experiments to demonstrate its usefulness and efficacy. This research makes two specific contributions: (i) a formal framework for modelling energy-aware cloud service allocation and (ii) a novel distributed allocation mechanism that integrates energy efficiency into the allocation rationale. The formal framework is based on game theory which provides formal means to analyse the most rational choice of actions for the interacting agents. We analyse optimal allocations based on the proposed framework and show that the utilitarian social welfare maximisation problem is *NP*-complete. While applying the *second-score auction* proposed by Che [9] to our setting, we design a distributed heuristic for this problem as an auction-based allocation mechanism, where rational SP agents are incentivised to truthfully reveal private information such as marginal costs.

The allocation mechanism is evaluated by means of multiagent simulation. This evaluation provides evidence of the mechanism's efficacy. Since the proposed approach constitutes a heuristic, it does not guarantee optimal solutions but provides approximations. Another limitation of the mechanism results from the distributed nature of the problem. There is no central coordinating agent in the setting considered. Therefore, the communication complexity is relatively high as compared to a centralised heuristic.

The remainder of this paper is structured as follows. In section 2, we describe the theoretical background of our research and discuss approaches in the extant literature. In section 3, we present the formal framework. In section 4, we describe the proposed allocation mechanism. Section 5 provides the experimental evaluation. Section 6 concludes the paper.

2 Related Work

Cloud service allocation is subject of inquiry in both multiagent systems and cloud computing research. We first introduce constructs of multiagent systems. Then we employ these constructs as a theoretical lens for reviewing the extant literature.

2.1 Multiagent Systems

Agent Definition. We adopt the definition of agents by Jennings: *"An agent is an encapsulated computer system that is situated in some environment and that is capable of flexible, autonomous action in that environment in order to meet its design objectives"* [10]. Apart from mere objects, agents have control over their internal state and their own behaviour, i.e., they possess *autonomy* over their choice of action. Thus, in line with [7], a software agent is specifically designed with the capability to act independently on behalf of its user or owner. That is, an agent is equipped with different goals of its user or owner and can discover by itself what needs to be done for the achievement of these objectives. In the following we investigate agent properties and clarify the implications for our work. These properties include (i) embeddedness in agent environments, (ii) autonomy, (iii) social ability, and (iv) deliberation and reactivity.

Agent Environment. From a technical perspective, the environment consists of anything an agent can percept through its sensors and act on through its effectors [11]. The emerging organizational context between agents defines the agents' relationship with each other [12]. Agent organizations provide a framework of constraints and expectations about the agents' behaviour with focus on decision making and action of specific agents [8]. The agents in our work are designed to take the *task-specific* roles of service providers and customers. The underlying *organizational structure* of this multiagent system describes the set of the agents' long-term responsibilities and interaction patterns [13].

Agent Autonomy. Agent autonomy implies that agents have their own goals, i.e., the agents' actions are driven by their own interests. Autonomy is always a relational concept [14]. An agent is autonomous in relation to the influence of other agents and the agent's environment, respectively. In our work, autonomous customer agents and service provider agents are equipped with utilitarian preferences reflecting their own delegated design objectives. Thus, our agents are *self-interested*, i.e., they are acting so to maximise their own utility. Hence, our game-theoretic approach describes the agents' autonomy as a relational concept where each individual agent can be viewed as a player in a negotiation game.

Agent Social Ability. The social ability of agents comprises the ability to communicate and cooperate with other agents. Agents own the ability to interact with other agents aiming at the satisfaction of their design objectives [7]. However, the agent's ability to merely exchange messages with other agents is insufficient from a goal autonomy perspective. Since other agents are autonomous themselves, they pursue their own (potentially conflicting) delegated objectives. Thus, as we cannot assume agents to share common goals, they must therefore *negotiate* and *cooperate* with other agents to achieve their goals [15]. Customer and service provider agents communicate by message passing. Messages are defined in the agents' interaction protocol and allow them to negotiate for the achievement of individual delegated objectives. In detail, our agents negotiate about price and energy consumption for the provision of cloud services.

Agent Deliberation and Reactivity. Deliberation is determining which state of the world is desirable to be achieved (goals) and how this state can be reached by performing appropriate actions. Reactivity denotes the agents' capability to sense their environments and to react upon recognised changes. In environments that involve dynamic changes (e.g., by other agents), agents must be reactive. However, this does not imply that the agent is capable of purely reactive behaviour, only. In contrast, agents reason about appropriate actions to respond to changed in the environment [7].

The environmental changes *indirectly* affect the agents' goals. Customer agents may react to changes in their environment by considering market competition and demand/supply ratios when requesting for services. Similarly, SP agents may react to service requests by strategically composing their bids based on market changes.

Multiagent Systems. Considering many real-world scenarios such as the market environment presented in this work, the design of problem solving processes requires more than a single agent, since there are *multiple loci of control*, i.e., there is a *decentralised nature of the problem* [10]. Hence, the agents' autonomy is of major importance for the applicability in scenarios with multiple, conflicting objectives. Thus it is necessary to move from the *micro level* of individual agents to the *macro level* of multiagent systems. Following Jennings [10], we use the multiagent-based software paradigm in order to (i) represent the distributed nature of the problem, (ii) enable multiple loci of control, and (iii) support competing interests of entities.

Both customer and SP agents act on behalf of independent enterprises that may be organisationally or geographically distributed. Centralized approaches employing some kind of global control are not applicable in this setting, as knowledge about data and resources is private information of the different stakeholders. Distributing control to multiple entities reduces the system's control complexity as a whole and causes individual components to be less coupled [10]. Further, both customer and SP agents are self-interested agents and pursue their individual business objectives as each agent tries to maximize its designated utility.

2.2 Energy-aware Service Allocation for Cloud Computing

Energy-efficient Cloud Computing. Most existing approaches addressing energy efficiency in cloud computing aim at the reduction of energy consumption in a single data centre. This goal is achieved by scheduling techniques and resource management for distributing thermal load or powering down server in times of low demand. Most of these approaches also fit for conventional data centres which are run and used by the same enterprise. [16] gives an overview of potential strategies to improve energy-efficiency in cloud environments. A theoretical evaluation of scenarios in which cloud computing has an advantage above local computing resources in terms of energy-efficiency is done by [17]. The authors evaluate the balance between computing, storage and network resources on a formal basis. [18] propose a low-energy scheduling model for the optimisation of energy-efficiency and use a welfare-maximisation approach. However, these approaches are limited to the optimisation of energy efficiency of a single enterprise. A global view on the energy efficiency of cloud computing and its impact in terms of carbon dioxide emissions is examined in [19].

Agent-based Cloud Service Allocation. Multiagent resource allocation is concerned with the way *resources* are distributed among multiple intelligent agents [6]. Though the notion of a "resource" can be used in its most general way, the resources we consider in this work are computational *services*. These services are *discrete*, i.e., they cannot be divided into smaller units, and *non-shareable*, i.e., a service cannot be used by multiple agents at the same time.

Bo and Lesser [5] investigate multiagent resource allocation across computational networks, where a set of selfish agents route traffic for individual users. Before user agents can route traffic through node entities, contracts between user agents and the participating nodes need to be established. The negotiation approach proposed is of distributed nature, i.e., agents act on behalf of themselves, and the corresponding resource allocation emerges from sequences of distributed negotiations. In addition to mutual contracting, nodes are allowed to decommit from existing contracts at a penalty cost. The authors investigate the relationship between stability and optimality of the network resource allocation game. Parkes et al. [20] present guidelines for the development of distributed allocation mechanism implementations based on the Vickrey-Clarke-Groves [21] mechanism. The aim of the approach is to distribute as much computation load as possible onto network nodes and thus help to determine a suitable allocation result. However, the proposed approach requires a 'center' entity which communicates with network nodes through a trusted channel and selects and enforces allocation outcomes.

In [22], a market-based approach is employed to efficiently allocate computing resources by means of an automated negotiation mechanism. Agents make bilateral contracts for resource leases and are given the ability to decommit from a contract by paying a negotiated penalty to the other contract party. However, the work above does not consider energy efficient allocation of computing resources.

3 Formal Framework

This section introduces a formal framework allowing us to calculate the optimal service allocation from a social welfare perspective. By assumption, switching off hosting servers results in the reduction of power consumption. Thus, each customer agent seeks to migrate all applications (henceforth services) running on the same physical machine to cloud-based SP agents. Since these services often depend on each other, they can only be migrated as *bundles* of services. However, customer agents will only move service bundles to cloud-based SP agents (and thus switch off physical servers) if cloud-based SP agents are able to offer a more energy-efficient way to execute the designated services. Thus, we assume each SP agent is aware of its energy efficiency. With regards to the standardized measurement of IT energy efficiency, we employ the widely accepted metric "performance-per-watt" (see [23] for a justification).

3.1 Agents and Services

To induce a migration decision, agents have to consider services executed on hosts, computational capacities, as well as energy efficiency specifications.

Agents. The set of customer agents is denoted by A_C and the set of SP agents is A_{SP}. The set of all agents is given by $A = A_C \cup A_{SP}$.

Services. The tuple of all services is denoted by $S = (s_1,s_2,...,s_n)$. Each service has a required computational capacity given by $w(s_k)$ with $k \in \{1,...,n\}$ measured in performance, i.e., computing operations per second. For example, a common performance metric, server side java operations per second (ssj_ops), is defined in SPECpower_ssj2008 [24]. We assume that different machines support the same performance metric.

Hosts. Each customer agent $i \in A_C$ owns a set of hosts (e.g., servers) denoted by $H_i = \{h_1^i,h_2^i,...,h_{m_i}^i\}$ with $m_i \in \mathbb{N}$. The average energy consumption rate (in watt) of each host is given by $E_i(h^i_l)$ with $l \in \{1,...,m_i\}$. Each host is defined by the binary tuple $h^i_l = (r_1,r_2,...,r_n)$, where $r_k = 1$ if service s_k is hosted on h_l^i and $r_k = 0$ if not. Due to the fact that i can only switch off its hosting hardware (and thus enhance its energy efficiency) once all services hosted by that particular hardware are migrated to SP agents, a customer agent i derives a valuation $V_i(h_l^i)$ for host h_l^i if all services hosted by h_l^i are allocated to SP agents.

Capacity. Each SP agent $j \in A_{SP}$ owns an infrastructure which provides a limited computational capacity denoted by W_j. Further, the energy efficiency of j is given by E_j (measured in performance/watt).

3.2 Agent Utility

The agents' individual utilities are influenced by their costs, payments for service provisioning, migration valuation and energy savings of specific hosts.

Cost. The cost of SP agent j for providing computational capacity for service s_k at energy consumption rate e_k is given by $c_j(w(s_k),e_k)$, where $c_j(w(s_k),e_k) = 0$ if no service is provided.

Payment. The payment from *i* to *j* for providing service s_k is represented by p_{ijk}. Since *i* seeks to migrate *all* services hosted by h_l^i, the total payment for host h_l^i is defined as the sum of payments of all services hosted by h_l^i, i.e. $P_i(h_l^i) = \sum_{j \in A_{SP}} \sum_{k=1}^{n} p_{ijk}^l$ where $p^l{}_{ijk}$ is the price for s_k hosted by h_l^i.

Allocation. An allocation is given by the tuple $X_{ijk} = (i, j, s_k, e_{ijk}, p_{ijk})$, where e_{ijk} denotes the contracted energy consumption rate of s_k. Let *X* denote the set of final allocations ("winning" allocations). The binary variable x_{ijk} is defined as $x_{ijk} = 1$ if $X_{ijk} \in X$, and $x_{ijk} = 0$ otherwise.

Utilities. The utilities of each customer agent *i* and SP agent *j* are given by

$$U_i = \sum_{j \in A_{SP}} \sum_{l=1}^{m_i} (V_i(h_l^i) - P_i(h_l^i)) x_{ijk}, \; U_j = \sum_{i \in A_C} \sum_{k=1}^{n} (p_{ijk} - c_j(w(s_k), e_k)) x_{ijk}.$$

3.3 Optimality of Service Allocations

We consider an allocation being optimal if the sum of all agents' individual utilities is maximised. Literature denotes this criterion as utilitarian social welfare, the standard performance metric in multiagent resource allocation. Such a metric is often used to measure the quality of an allocation with regards to the system as a whole [6]. The utilitarian social welfare is calculated as

$$\begin{aligned} W &= \sum_{i \in A_C} U_i + \sum_{j \in A_{SP}} U_j \\ &= \sum_{i,j \in A} \left(\sum_{l=1}^{m_i} (V_i(h_l^i) - P_i(h_l^i)) x_{ijk} + \sum_{k=1}^{n} (p_{ijk} - c_j(w(s_k), e_k)) x_{ijk} \right) \qquad (1) \\ &= \sum_{i,j \in A} \left(\sum_{l=1}^{m_i} V_i(h_l^i) x_{ijk} + \sum_{k=1}^{n} c_j(w(s_k), e_k) x_{ijk} \right). \end{aligned}$$

Note that the payments do not appear in this equation since these simply redistribute the wealth between the agents. In the following we are interested in finding the optimal set of services that needs to be allocated to SP agents such that the social welfare is being maximized, i.e.,

$$W^* = \max_x \sum_{i,j \in A} \left(\sum_{l=1}^{m_i} V_i(h_l^i) x_{ijk} - \sum_{k=1}^{n} c_j(w(s_k), e_k) x_{ijk} \right) \qquad (2)$$

$s.t.$

$$\forall j \in A_{SP} : \sum_{i \in A_C} \sum_{k=1}^{n} w(s_k) x_{ijk} \leq W_j . \tag{3}$$

Theorem 1. The computation problem for the socially optimal allocation is *NP*-complete.

Proof. NP membership is easy, since for any given solution to our allocation problem the calculation of the resulting social welfare and the verification of the capacity constraints defined in (3) can obviously be done in polynomial time. For hardness, we define a reduction from the 0-1 Knapsack problem which is known to be *NP*-complete [25]. The 0-1 Knapsack problem is defined as follow. Let *U* be a finite set and let $u \in U$. Further, $g(u)$ denotes the weight and $b(u)$ the valuation of element *u*, and *K* is a positive integer. Further, $a(u)$ is a non-negative integer for each $u \in U$. Now maximize $\sum_{u \in U} a(u) b(u) \, s.t. \sum_{u \in U} a(u)\, g(u) \leq K$ (cf. [26]). Any instance of the 0-1 Knapsack problem can be reduced to our optimization problem as follows. Let $m_i = n = 1$ in equation (2), i.e., each customer agent in A_C has exactly one service bundle ($l = 1$) containing exactly one service ($k = 1$), and let $l = 1$, i.e., one SP agent only, then we can rewrite equation (2) as

$$W^* = \max_x \sum_{i \in A_C} \big(V_i(h_1) - c_1(w(s_1), e_1) \big) x_{i11} \tag{4}$$

with constraint $\sum_{i \in A_C} w(s_1) x_{i11} \leq W_1$. Further, set $A_C = U$, $x_{i11} = a(u)$, $V_i(h_1) - c_1(w(s_1),e_1) = b_1$, $w(s_1) = g(u)$ for all $u \in U$ and $W_1 = K$. Obviously, this reduction can be done in polynomial time. Hence, each SP agent's optimization problem and therewith the utilitarian social welfare maximization problem is *NP*-complete.

3.4 Illustrative Example

We provide an example to illustrate the formal framework. This examples considers two customers c1 and c2, three service providers sp1, sp2, and sp3, and a set of services $S = \{s_1, s_2, s_3\}$ as depicted in figure 1. Each SP has an energy efficiency E_i in performance per watt, a capacity W_j in the performance metric ssj_ops, and a cost in € to provide its services c_j with $j = \{1,2,3\}$. On the demand side, both c1 and c2 own one host h_1 and derive a valuation $V_i(h_1)$ with $i = \{1,2\}$, if that particular host can be switched off. Hence, c1 requests for computing capacities $w(s_1)$ and $w(s_2)$ and c2 for $w(s_3)$. Table 1 shows all possible service allocations denoted by a1 to a16 and presents the social welfare achieved by each allocation. For the calculation, customer valuation and SP cost are used as *per computing unit*, e.g., the valuation per computing unit of c1 equals 85.5/450 = 0.19. Obviously, allocation a1 with 85.5 + 41.4 − 26.0 − 27.5 − 27.6 = 45.8 maximizes the social welfare for the given setting.

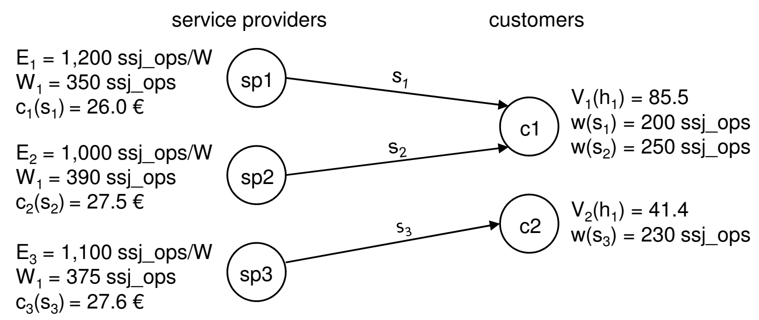

Fig. 1. Example 1 with two customers and three service providers

Table 1. Possible allocations and social welfare for example 1

Service request		**a1**	**a2**	**a3**	**a4**	**a5**	**a6**	**a7**	**a8**	**a9**	**a10**	**a11**	**a12**	**a13**	**a14**	**a15**	**a16**
c1	s1	sp1	sp2	sp3	sp1	sp2	sp3	sp1	sp1	sp2	sp2	sp3	sp3	-	-	-	-
	s2	sp2	sp1	sp2	sp3	sp3	sp1	sp2	sp3	sp1	sp3	sp1	sp2	-	-	-	-
c2	s3	sp3	sp3	sp1	sp2	sp1	sp2	-	-	-	-	-	-	sp1	sp2	sp3	-
Social welfare		45.8	44.8	45.5	45.6	45.0	45.1	32.0	29.5	31.0	33.5	29.0	34.0	11.5	16.1	13.8	0.0

4 Allocation Mechanism

Since finding the optimal service allocation is a computationally infeasible problem, we propose a distributed heuristic for energy-aware cloud service allocation in order to approximate agents' decision-making. We employ a multi-attribute combinatorial procurement auction where multiple customer agents request for bundles of energy-efficient cloud services from strategic service provider agents. At first, we introduce a formal auction model and then describe an allocation protocol that is *incentive-compatible* for SP agents. An auction protocol is called *incentive-compatible* if it is each SP agents' dominant strategy to declare its true preference over the requested services regardless of other SP agents' actions [27].

4.1 Auction Model

The agents' participation in the auction requires service requests and bids as well as a score combining price with energy consumption.

Request. Each customer agent $i \in A_C$ requests for service bundles defined by $R_i = (R_1^i, R_2^i, \ldots, R_{m_i}^i)$, where each service bundle R_l^i is represented by the binary request bundle of the form $R_l^i = (r_1, r_2, \ldots, r_n)$, i.e., $r_k = 1$ if service s_k belongs to the requested bundle and $r_k = 0$ if not.

Score. Let e_k denote the energy consumption rate of service s_k and p_k the price for service provisioning. For each tuple (e_k, p_k) we adopt the notion of [9] and define a quasi-linear scoring function by

$$\sigma_i(s_k, e_k, p_k) = \alpha_i\, w(s_k)/e_k - \beta_i p_k, \tag{5}$$

where $\alpha_i, \beta_i \in \mathbb{R}^+$ are used for scaling purposes. The scoring function is assumed to be publicly known to all SP agents at the start of bidding. As in [9], we consider the second-score auction where the winning SP agent is obligated to match the second highest score. However, in meeting this score, the winner may choose any tuple (e_k, p_k). As an example, let SP agent A's bid for service s_1 be (5, 0.2) and B's bid (6, 0.15), and the scoring function is such that $\sigma_i(s_1, 5, 0.2) = 10$ and $\sigma_i(s_1, 6, 0.15) = 9$. Here, SP agent A wins the auction and is free to choose any tuple (e', p') such that $\sigma_i(s_1, e', p') = 9$ is fulfilled.

Bids. Each SP agent j's bid is given by the n-tuple $B_j = (B^j_1, B^j_2, \ldots, B^j_n)$, where each B^j_k is a tuple of the form $B^j_k = (\sigma_i(s_k, e_k, p_k), e_k)$ with the score $\sigma_i(s_k, e_k, p_k)$, and the promised energy consumption rate e_k for providing service s_k. If j does not bid for service s_k, we set $B^j_k = (0,0)$.

4.2 Auction Protocol

We consider the widely used one-shot protocol described in [28]. Further, our protocol is assumed to be *individually rational*, i.e., no agent will commit to a contract for service provision if its cost exceeds the payment received. This is natural as no agent is willing to participate in an auction where it spends more than what it earns. In the following we describe the individual steps of the protocol.

- Each customer agent i asks for bids from SP agents by announcing its request $R_i = (R^i_1, R^i_2, \ldots, R^i_{mi})$ to all SP agents.
- For each service s_k in i's request bundles R^i_l, SP agents send their binding bids B_i to customer agents.
- Customer agent i evaluates the received bids and informs SP agents about the acceptance of their bids by transferring the second highest score.
- The winning SP agents calculate the price based on the second highest score. A contract is established and second score payments are transferred. For each rejected bid, no contract is established.

Since customer agent i only derives non-zero valuations for complete service bundles, it is individually rational for i to establish contracts if and only if it has received acceptable bids for *each* service in the current bundle. Once this is the case, i selects the SP agent with the highest score for each requested service. The winning SP agent is then informed about the acceptance of its bid and the value of the auction's second highest score denoted by $\sigma_{i,2}$. Based on the scoring function defined in (5), the winning SP agent chooses an energy-price tuple (e', p') such that $\sigma_i(s_k, e_k, p_k) = \sigma_{i,2}$. A contract is established for each service s_k within the bundle specifying the price p' and the energy consumption e' for service s_k.

Note that SP agents may choose to lie when bidding, i.e., they may systematically overvalue their offered energy efficiency and base the transmitted score defined in (5) on costs that are higher than its true valuation. In the following we apply [9] and show that this protocol is incentive-compatible for SP agents, i.e., it is each SP agent's dominant strategy to declare its true preferences for service allocations. Thus, each individually rational SP agent will always calculate its bidding score on its true marginal cost.

Theorem 2. The proposed auction protocol is incentive-compatible for SP agents, where each SP agent's bid for providing service s_k is calculated using $p_{ijk} = c_j(w(s_k),e_{ijk})$.
Proof. Applying lemma 1 from [9], we set $q = e_k, s(q) = \alpha_i w(s_k)/e_k$, $c(q,\theta) = \beta_i c_j(w(s_k), e_k)$ and $q_s(\theta) = e_{ijk}$. Then $e_{ijk} = \arg\max(\alpha_i w(s_k)/e_k - \beta_i c_j(w(s_k),e_k))$ which reduces the two-dimensional auction to a one-dimensional problem. Hence, proposition 1 (ii) in [9] yields $p_{ijk} = \beta_i c_j(w(s_k),e_{ijk})$. Consequently, SP agents' dominant strategy consists in calculating their bidding scores based on their marginal costs $c_j(w(s_k),e_{ijk})$.

From this result it becomes clear that SP agents will always calculate their score based on their true marginal cost, no matter on how other SP agents choose their score. With this in mind, we can deduce the following corollary.

Corollary 1. For a single customer agent request, the proposed protocol results in an optimal service allocation.
Proof. By theorem 2, it is individually rational for each SP agent to calculate its bid for providing service s_k by using its marginal cost $c_j(w(s_k),e_k)$. That is, only SP agents that offer minimal cost will be contracted by the customer agent. As customer agent's valuation for each service bundle h^i_l is given by $V_i(h^i_l)$ and costs $c_j(w(s_k),e_k)$ are minimized, the resulting service allocation is optimal.

5 Evaluation

This section provides the experimental evaluation of the proposed allocation mechanism. We describe the experimental setup, report the results, and discuss the findings.

5.1 Experimental Setup

The parameters used in our experiments are based on pricing models of Amazon in 2012 and latest power and performance information found in the SPECpower standard. According to the Amazon EC2 User Guide and SPECpower, the performance of a Dell PowerEdge R720 approximately corresponds to the computing capabilities of 88 EC2 small instances [24]. Hence, on average, one million ssj_ops cost 6.75 € and consume 186 Watt power. Thus, in our setup, costs are normally distributed with $\mu_c = 6.75$ and standard deviation $\sigma_c = 0.1$. Based on Amazon and SPECpower data, the energy efficiency of SPs and customer agents' hosts are assumed to follow a normal distribution with expectation $\mu_E = 5365$ ssj_ops/W and $\sigma_E = 268.2$. We assume that SP agents' costs correlate positively with its energy efficiency. The following random numbers were generated using normal distribution:

- $X_{c,j} \sim N(\mu_c; \sigma_c^2)$ refers to SP agent *j*'s fixed cost component incurred for service provisioning,
- $X_{E,j} \sim N(\mu_E; \sigma_E^2)$ refers to SP agent *j*'s energy efficiency,
- $c_j = X_{c,j} (X_{E,j} / \mu_E)$ refers to SP agent *j*'s cost for one million ssj_ops,
- $X_{c,j} \sim N(\mu_c; \sigma_c^2)$ refers to customer agent *i*'s fixed valuation component,
- $V_i(h_l^i) = X_{c,j} + \mu_E / X_{E,j})$ refers to customer agent *i*'s valuation per computing unit for host h_l^i.

The number of hosts per customer agent is Poisson distributed with an expectation $\lambda_k = 10$. The number of services per host is also Poisson distributed with an expectation $\lambda_s = s$ and each service is taken by uniform distribution from a set of 10 services with performance ranging from 0.3 to 2.1 million ssj_ops. The number of SP agents is set to 5 as we assume there exists a relatively small number of large cloud SPs on the market, e.g., Amazon and IBM. SP agents' computational capacities are generated by normal distribution such that demand and supply are balanced at 50 customer agents. Each experiment was performed 100 times. The optimal values of the allocations' social welfare are computed using CPLEX (a commercial optimization software by IBM).

5.2 Results

Both figures 2 and 3 show the average behaviour of the utility ratio (agents' social welfare relative to optimal utility) as a function of the number of customer agents where energy efficiency and price are weighted using three different ratios. Different weights can be set by means of α_i and β_i as defined in the scoring function (5) and are equal for all customer agents. In the following, energy-price weighting ratios are simply written in the form energy:price (e.g., 1:1 means energy and price are weighted equally).

Figure 2 shows a setting where higher weights are placed on energy efficiency. The utility ratio at 1:1 monotonically increases in the interval [2,44] until the maximum close to 1.0 is reached at 44 customer agents. The utility ratio then decreases again and seemingly converges to 0.9. Similarly, the utility ratio for 3:2 also increases in the interval [2,44], but starts at a lower utility ratio as compared to the function with 1:1, showing a higher slope. It then decreases and indicates a convergence to the utility ratio of 0.9. Further, the utility ratio for 4:1 is significantly low for a small number of customer agents while it increases at a high slope until the maximum close to 1.0 is reached at 44 customer agents. Hence, it finally decreases and reaches a utility ratio of 0.9.

Figure 3 shows a setting where higher weights are placed on price rather than on energy efficiency. The utility ratio with 1:1 is the same as the one displayed in figure 2. At 2:3 the utility ratio dominates the one for 1:1 and increases monotonically in the interval [2,44] until the maximum close to 1.0 is reached at customer agent number 44. Similarly, the weights 1:4 cause the utility ratio to exceed all other ratios while constantly remaining close to 1.0 in the interval [2,44]. It then decreases for a growing number of customer agents and reaches a utility ratio of 0.9.

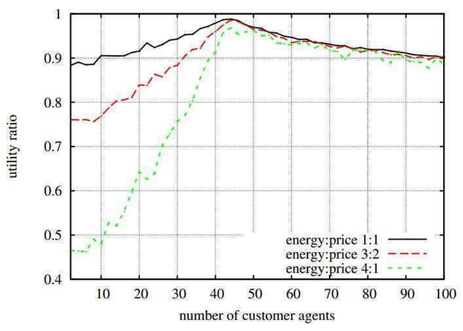

Fig. 2. Utility ratio as function of customer number with higher weights on energy.

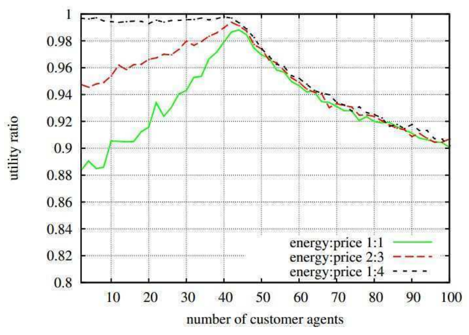

Fig. 3. Utility ratio as function of customer number with higher weights on price.

5.3 Discussion

Our experiments demonstrate the influence of energy-aware customer agents on the agents' social welfare and provide evidence for the usefulness and efficacy of the proposed auction mechanism. In the interval [2,44] all "energy-aware" utility ratios displayed in figure 2 increase monotonically. This result demonstrates the impact of the changing demand/supply-ratio with an increasing number of customer agents: Within the interval [2,48], on average, service supply is greater than the requested service demand. Hence, on a market with excess supply, energy-aware customer agents easily satisfy their demand and choose energy-efficient services. The incurring costs of SP agents, however, are high when providing services of high energy efficiency. Therefore, customer agents that overvalue energy efficiency as compared to price will purchase services from expensive SP agents. Hence, capacities of cheap SP agents remain unused.

Since the social welfare decreases at higher SP agent costs and unchanging customer agent valuation, both utility ratios for 4:1 and 3:2 are low for a small number of customer agents but increase until the market is saturated (on average, this happens at 50 customer agents). That is, increasing weights for energy efficiency result in higher costs for service provisioning and hence in lower utility ratios. When the market is balanced, the utility ratios reach its maximum for all energy-price weights. The fact that the utility ratios then decrease within the interval [46,100] is intuitive: Once services for sale are beginning to be scarce, resource competition is high and the mechanism can only achieve a high social welfare if services of high valuation can be migrated to SP agents. In such situations, customer agents are willing to forgo their energy-price preferences in order to obtain their valuations. Further, with a growing number of customer agents, the success rate of the mechanism decreases as it becomes more difficult for competing customer agents to migrate their services. Hence, the utility ratios for all energy-price preferences reach 0.9.

Experiments for the setting with "price-aware" customer agents as displayed in figure 3 confirm a rather trivial intuition: When placing higher weights on the price and lower weights on energy efficiency (e.g., 1:4), the mechanism achieves a constant maximal utility ratio close to 1.0. Obviously, in a market with low competition, customer agents acquire services from SP agents with lowest cost while paying only little attention to energy efficiency. Once resources are being scarce (i.e., beyond 50 customer agents), the high competition among customer agents causes the utility ratio to finally reach 0.9 for all considered energy-price preferences.

6 Conclusion

This work presents a formal framework for modelling energy-aware cloud service allocation and proposes a distributed allocation mechanism that integrates energy efficiency into the allocation rationale. Customers and cloud SPs are represented as software agents that autonomously negotiate service agreements based on a set of different energy-price preference ratios. We employ game theory to analyse optimal service allocation and show the *NP*-completeness of the underlying utilitarian social welfare maximisation problem. We develop a distributed heuristic for the allocation problem. The proposed allocation mechanism, a second-score auction protocol, is shown to be incentive-compatible for SP agents. We evaluate this mechanism by means of multiagent simulation. The current formal framework is limited to two non-functional properties of services, i.e., energy efficiency and performance. In our future work, we plan to include additional properties such as response time and availability.

Acknowledgements. This work has been supported by (1) the project MIGRATE!, funded by the German Federal Ministry of Economics and Technology (BMWi, FKZ 01ME11052), and (2) the eHealthMonitor project (http://www.ehealthmonitor.eu), funded by the European Commission under contract FP7-287509.

References

1. Weiss, A.: Computing in the clouds. netWorker 11 (4), 16–25 (2007)
2. Weinhardt, C., Anandasivam, A., Blau, B., Borissov, N., Meinl, T., Michalk, W., Ster, J.: Cloud-computing. Wirtschaftsinformatik 51, 453–462 (2009)
3. The green grid consortium, http://www.thegreengrid.org (2011)
4. Sen, A.K.: Collective Choice and Social Welfare. Holden-Day, Michigan (1970)
5. Bo, A., Lesser, V.: Characterizing contract-based multi-agent resource allocation in networks. IEEE Sys. Man. Cybern. 40, 575–586 (2010)
6. Chevaleyre, Y., Dunne, P.E., Endriss, U., Lang, J., Lemaitre, M., Maudet, N., Padget, J., Phelps, S., Rodriguez-Aguilar, J.A., Sousa, P.: Issues in multiagent resource allocation. Informatica 30, 3–31 (2006)
7. Wooldridge, M.: An Introduction to MultiAgent Systems. 2nd edn. John Wiley & Sons, Chichester (2009)
8. Bond, A.H., Gasser, L. (eds.): Readings in Distributed Artificial Intelligence. Morgan Kaufmann Publishers, San Mateo (1988)

9. Che, Y.K.: Design competition through multidimensional auctions. Rand J. Econ. 24, 668–680 (1993)
10. Jennings, N.: On agent-based software engineering. Artif. Intell. 117, 277–296 (2000)
11. Russell, S.J., Norvig, P.: Artificial Intelligence: A Modern Approach. 2nd edn. Prentice Hall, New Jersey (2003)
12. Ferber, J., Gutknecht, O.: A meta-model for the analysis and design of organizations in multiagent systems. In: Proceedings on the Third International Conference on Multi-Agent Sytstems (ICMAS-98) (1998)
13. Durfee, E.H., Lesser, V.R., Corkill, D.D.: Coherent cooperation among communicating problem solvers. IEEE Transactions on Computers C-36, 1275–1291 (1987)
14. Castelfranchi, C., Falcone, R.: Founding Autonomy: The Dialectics Between (Social) Environment and Agent's Architecture and Powers. In: Nickles, M., Rovatsos, M., Weiss, G. (eds.): Agents and Computational Autonomy. LNCS, Vol. 2969, pp. 40-54. Springer, Heidelberg (2004)
15. Castelfranchi, C.: Modelling social action for AI agents. Artif. Intell. 103, 157–182 (1998)
16. Berl, A., Gelenbe, E., Di Girolamo, M., Giuliani, G., De Meer, H., Dang, M.Q., Pentikousis, K.: Energy-efficient cloud computing. Comput. J. 53, 1045–1051 (2009)
17. Baliga, B.J., Ayre, R.W.A., Hinton, K., Tucker, R.S.: Green cloud computing: Balancing energy in processing, storage and transport. In: Proceedings of the IEEE 99, pp. 149–167 (2010)
18. Bodenstein, C., Hedwig, M., Neumann, D.: Low-energy automated scheduling of computing resources. In: 1st IEEE/ACM Workshop on Autonomic Computing for Economics (2011)
19. Garg, S.K., Yeo, C.S., Buyya, R.: Green cloud framework for improving carbon efficiency of clouds. In: Jeannot, E., Namyst, R., Roman, J. (eds.): Euro-Par 2011. LNCS, Vol. 6852, pp. 491–502. Springer, Heidelberg (2011)
20. Parkes, D., Shneidman, J.: Distributed implementations of Vickrey-Clarke-Groves mechanisms. In: Proceedings of the 3rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS'04), pp. 261–268. ACM, New York (2004)
21. Vickrey, W.: Counterspeculation, auctions, and competitive sealed tenders. J. Financ. 16, 8–37 (1961)
22. Bo, A., Lesser, V., Irwin, D., Zink, M.: Automated negotiation with decommitment for dynamic resource allocation in cloud computing. In: Ninth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'10), pp. 981–988 (2010)
23. Sharma, S., Hsu, C., Feng, W.: Making a case for a green500 list. In: Proceedings of the Workshop on High-Performance, Power-Aware Computing (2006)
24. SPEC: Standard performance evaluation corporation, http://www.spec.org (2012)
25. Karp, R.: Reducibility among combinatorial problems. In: Miller, R., Thatcher, J. (eds.): Complexity of Computer Computations. Plenum Press, New York (1972)
26. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman, New York (1979)
27. Mas-Colell, A., Whinston, M.D., Green, J.R.: Microeconomic Theory. Oxford University Press (1995)
28. Rubinstein, A.: Perfect equilibrium in a bargaining model. Econometrica 50, 97–109 (2006)