Preventing Collusion in Cloud Computing Auctions

Shunit Agmon[1], Orna Agmon Ben-Yehuda[1,2], and Assaf Schuster[1]

[1] Technion—Israel Institute of Technology, Haifa 3200003, Israel
{shunita,ladypine,assaf}@technion.ac.il
[2] University of Haifa, Caesarea Rothschild Institute for Interdisciplinary Applications of Computer Science

Abstract. Cloud providers are moving towards auctioning cloud resources rather than renting them using fixed prices. Vickrey-Clarke-Groves (VCG) auctions are likely to be used for that purpose, since they maximize social welfare—the participants' aggregate valuation of the resources. However, VCG auctions are prone to collusion, where users try to increase their profits at the expense of auction efficiency. We propose a coalition formation mechanism for cloud users that helps both users and providers. Our mechanism allows the auction participants to collaborate profitably while also maintaining the auction's resource allocation efficiency. Our experiments show that when using our mechanism, participants' mean profit increases by up to 1.67x, without harming the provider's allocation efficiency.

Keywords: Cloud, Auctions, Collusion

1 Introduction

Cloud computing provides flexibility to clients by allowing them to pay per use for the rental of services and VMs. Renting reduces the waste of prepurchased but unutilized hardware [14]. Recently, cloud computing has been moving towards the more economical Resource-as-a-Service model (RaaS) [10, 12]: instead of horizontal scaling (renting more VMs), RaaS clouds enable vertical scaling—renting more resources (such as CPU, RAM, and I/O resources) for a few seconds at a time, at sub-second granularity. For example, CloudSigma charges separately for CPU, RAM, SSD storage, and data transfer, and it adjusts burst prices every few minutes [5]. Amazon Web Services (AWS) [1], Azure [4], and Google Cloud Platform [6] all offer a pay-as-you-go pricing method. AWS Lambda [2] and Azure Functions [3] allow uploading code and paying for computing time only when the code is triggered to run.

RaaS systems use economic mechanisms, such as auctions, to allocate resources [13, 22, 44]. AWS EC2 spot instances [11], Alibaba Cloud spot instances [7], and Packet spot market [8] are examples of auctions in horizontal elasticity. We predict that auctions will be deployed in vertical elasticity as they are now deployed in horizontal elasticity.

The Vickrey-Clarke-Groves (VCG) auction [43, 20, 24] is well-suited for this purpose. VCG auctions maximize the social welfare: the aggregate valuation of all users of the resource allocated to them. They allocate resources first to the users who value them most, and thus enable getting the most out of the machine. VCG auctions are already used by Facebook to allocate ad spaces [32] and have been used by Google for contextual ads since 2012 [42]. They have also been shown suitable for network bandwidth allocation [27].

Ginseng systems are examples of VCG auctions used for resource allocation (for RAM [13] or cache [22]). In this work we focus on Ginseng for RAM (referred to as Ginseng from this point on). Ginseng [13] consists of a market-driven RAM allocation mechanism called Memory Progressive Second Price (MPSP) that resembles a VCG auction. In Ginseng, guests run economic agents who bid for resources auctioned by the host. Thus, the mechanism incentivizes selfish, rational agents, who only care about their own profit, to bid with their true valuation of the RAM. Ginseng was shown to achieve up to 16% improvement in guest benefit from RAM allocation and up to 43% improvement from cache allocation, compared to state-of-the-art approaches. Since the number of auction participants is bound by the number of VMs on a single physical machine, the host computation time is not a bottleneck. For example, for 24 guests, the computation takes less than a second.

However, VCG auctions are not perfect. To maximize the social welfare, they incur additional costs to the users, possibly hindering profitability. VCG auctions, especially in repeated settings, are also not collusion-proof [17, 18, 21, 23, 41, 28, 30, 36, 38, 40]. Colluding to increase profit may reduce social welfare [21], e.g., by bid rotation [31, 15, 16, 35] or sub-optimal redistribution of the goods [21, 36]. In a cloud environment, goods can only be transferred with the host's consent, so a collusion scheme involving resource transfer is impossible, but other forms of collusion are possible. For example, consider a cloud machine with 10 GB of RAM, and two VMs running memory-heavy applications, requiring the entire available RAM. Suppose one VM owner (Alice) values the RAM at 15¢ per hour, while the other (Bob) values it at 10¢ per hour. Alice and Bob can discover who the other VM belongs to [37], and they can agree on a collusion scheme. Since agents care only about their profits and are not exposed to each other's private information, there is no guarantee that they will agree on an efficient scheme, where Alice gets the RAM. Instead, they might agree that Bob gets the RAM and compensates Alice 7¢ per hour for not bidding. Both their profits increase, but the social welfare drops from 15¢ to 10¢ per hour.

We propose a platform for collaboration among guests that will increase their profits, thus reducing their incentive to collude, without changing the auction efficiency and the social welfare. In this model, guests can ask the host to consider them as a single guest when computing their bill. Since MPSP is based on exclusion-compensation [27], where guests pay for the damage they cause others, they would pay less if they are billed together. The host acts as a trusted third party, since guests already share their (private) valuations with the host in the auction. They tell the host how they want to share the discount (in terms of profit parts). The host calculates the reduced bills accordingly. A host might support such interactions if its main revenue is from a base payment rate, as in Ginseng, where each guest rents a base amount of the resource for a fixed price. A host also might support such interactions in private clouds, where the only goal is to maximize social welfare. This mechanism does not significantly increase the computational load on the host: computing a bill for a coalition is computationally equivalent to computing the bill for a single guest.

Our contribution is the proposal, implementation, and evaluation of a guest coalition mechanism that does not harm social welfare and which RaaS hosts have an incentive to support. It allocates resources efficiently while lowering guests' costs, thus reducing their incentive to collude in a harmful manner. The economic mechanism can thus be used for its original purpose: optimal resource allocation, which leads to optimal hardware use. The implementation was released as free software [9].

2 Simplified Memory Progressive Second Price Auction

We begin by describing a representative VCG-like resource auction, called Simplified Memory Progressive Second Price (SMPSP). It resembles bandwidth auctions [27, 29], and is identical to the MPSP auction used by Ginseng [13] when guests have monotonically increasing, concave functions, consistent with diminishing returns (which are common in auction schemes [27, 29]). The SMPSP auction is identical to repeating the auction proposed by Maillé and Tuffin [29] when the valuation function is approximated by using a single point. SMPSP is also used by Movsowitz *et al.* [34]. As in [29], this auction converges to approximately optimal social welfare, and therefore approximately optimal allocation.

SMPSP is a repeated auction, performed in rounds, each composed of steps. In each SMPSP round, the host first announces the free amount of resource for rent. Each guest i responds with a bid: a pair (p_i, q_i), where p_i is the unit price the guest is willing to pay to rent a resource quantity q_i (the *bid quantity*). The host collects all bids. Guests are sorted in decreasing order by their unit prices and allocated their bid amount (or the free amount left). Bills are calculated according to the exclusion compensation principle. Let q'_i denote the amount allocated to guest i, and let q''_j denote the amount guest j would have received if guest i did not exist. Then the bill guest i would pay is given by

$$B_i = \frac{1}{q'_i} \sum_{j \neq i} p_j (q''_j - q'_j). \tag{1}$$

Finally, guests are notified of the new allocation and the host redistributes the resource.

Throughout this work, we assume that allocation efficiency is more important to the provider than the guest auction payments. This assumption always holds in private clouds, and sometimes in public clouds. In public clouds, the available resources usually suffice for all the guests. When there is temporary resource pressure, the auction mechanism serves as a bridge solution until the problem is solved: either by the client purchasing reserved resources, or by the provider migrating the client to a different physical machine. Therefore, in public clouds as well, the auction payments are not the main source of revenue, but auction efficiency is crucial, as it allows the provider more time to handle the migration.

We focus on the case where the resource does not suffice for all the guests, as sometimes happens in spot instances [1, 7, 8]. Otherwise, SMPSP bills are 0 and the coalitions are unnecessary.

3 Related Work

In VCG and VCG-like auctions, guests may collude in various ways, most of which reduce the social welfare by preventing optimal allocation. One possible collusion scheme is a secondary resource market, conducted without the host's awareness [23, 28, 31]. In this scheme, a bidding ring is formed: a special trusted agent or third party acts as the ring center and holds a knockout auction among ring members to decide on the winner and on money transfers. In another work [17] the ring center is replaced by complete knowledge of the ring members' valuations, and the designated winner passes the won goods to other colluders. Neither the existence of a trusted specific agent nor the complete knowledge of valuations is practical in our case, due to the lack of trust between guests. In addition, in our model goods can only be transferred with the host's consent.

Another possible scheme is price shading, where one agent bribes another for falsely reporting its bid price or reducing it to zero [21, 36]. This scheme can harm the allocation efficiency: there is no guarantee that the briber is the agent with the higher valuation, and, when bills are artificially lowered, winning guests are incentivized to increase their bid quantity beyond their original request (without collusion). Since the original allocation was approximately optimal, the change will harm its efficiency.

Bid rotation [31, 15, 16, 35, 18, 40] can be seen as a private case of price shading in repeated auctions, where agents take turns falsely reducing their bid price to 0 (effectively not participating in the auction). In addition to harming the social welfare, when the auctioned resource is RAM, this scheme also harms the agents themselves: frequently passing RAM from one agent to another reduces its utility, as demonstrated by Movsowitz *et al.* [34].

Krause *et al.* [26] present coalition formation in a task oriented domain: agents form a group to perform a task and must decide on a division of the profit. Several division strategies are proposed: equal division of profits, division proportional to contribution, or by an algorithm that finds a stable division—such that there is no group of agents with an incentive to defect. An agent can choose to forgo some profit in each of the division strategies. Simulations showed that equal division with a compromise of 20% yields the highest profits. Implementing these findings in our model is left for future work. Chatterjee *et al.* [19] present bidding rings as a coalition formation bargaining game, but assume bidders have complete information regarding their valuations.

The negative effects of collusion between auction participants and the recommended auctioneer responses have been widely studied. Some works study the effects of money transfers [31], cooperation enforcement types [30], and auction types [41, 30] on viable collusion schemes. Second price auctions were found to be more amenable to incentive-compatible collusion schemes than first price. Recommended auctioneer responses include reserve prices [23, 31], ceiling prices [25], and withholding information about the value of the goods [38].

Movsowitz *et al.* [33] review cloud attacks that can be applied in the RaaS model and various ways to defend against them. Movsowitz *et al.* [34] study economic attacks that are unique to the RaaS model, taking advantage of the economic mechanisms used in RaaS clouds.

Our collaboration mechanism does not involve agents changing their bids and does not assume any trust between them. If the attackers are rational, i.e., profit-driven, our mechanism makes both attacks and collusion less appealing, by reducing the profit to be gained from them.

4 Negotiation Protocol

Agents form a coalition that the host considers as a single agent when computing their bill. Agents are still charged separately, but the sum of their bills is lower. They negotiate the division of the additional profit.

The agents negotiate after each auction round's results are announced, before submitting a new bid (see Fig. 1). Each agent can negotiate with all others, but can only create a coalition with one other agent/coalition in each round. This is a design choice intended to simplify the protocol. It could easily be changed so that each auction round contains multiple negotiation rounds.

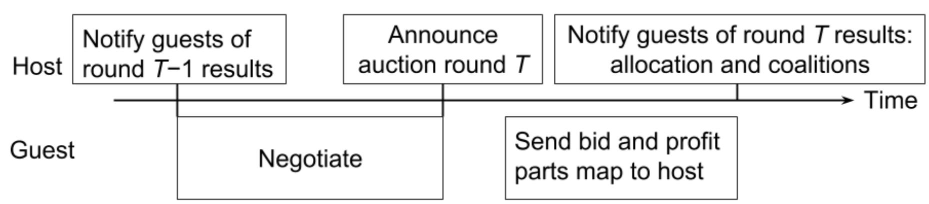


Fig. 1. Coalition Protocol: Auction Timeline

4.1 Negotiation between Two Agents

In the base case of the protocol, two agents negotiate the creation of a coalition. Each protocol step consists of an offer, followed by a positive response (*accept*), a rejection or a counter-offer. An offer is a map from agents to *profit parts*—positive fractions of the profit each agent receives, whose sum is 1.

If a coalition is formed, both agents send the host this map. The host uses it to compute their bills as follows: Let T_C denote the bill for the entire coalition, calculated as if agents in the coalition were one combined agent. If agent i gets amount q'_i of the resource, let $B_i(q'_i)$ denote i's undiscounted bill for it. Finally, let f_i denote the fraction of the coalition's discount that i should get. Then the new bill for coalition member i is

$$bill_i = B_i\left(q'_i\right) - f_i\left(\sum_{j\in C} B_j\left(q'_j\right) - T_C\right). \tag{2}$$

An agent's bill may be negative. For example, if i had no bill before joining the coalition, i.e., $B_i = 0$, and i and j are the only agents in the system, then the total bill of their coalition is 0. But since i has a positive profit part, the host transfers money from j to i.

Upon notification of an auction round's results, the host also notifies each agent of its actual bill, the bill it would have paid outside the coalition, and the profit parts map (to confirm the coalition's validity). The agent can then compute its profit from the coalition in this round.

4.2 Negotiating a Coalition of Coalitions

In the following rounds, a two-agent coalition can grow by negotiating with another agent/coalition. The leader of the larger of the two coalitions leads the joint coalition in further negotiations. If coalition sizes are equal, the agent whose proposal was accepted is the leader. If approached, a non-leader defers to the leader. Only the agents are aware of the chosen leader: this role is irrelevant for the host.

This design decision encourages coalition growth, since the leader of the larger coalition has successfully negotiated before. Moreover, if only agents that were allocated their full bid quantity are allowed to participate in coalitions (see §5.1), that leader is likely to be allowed to participate in the next auction round as well. For same-size coalitions, choosing the accepting party as the leader may mean choosing a leader whose proposals have been rejected, and thus are likely to be rejected in the future, preventing the coalition's growth.

Two leaders negotiate the merger of their respective coalitions similarly to the two-agent protocol. First they report their coalition sizes in a preliminary protocol step. Leaders normalize the sent or received offers by coalition sizes (as explained below). After negotiating the profit parts as in the two-agent case (see §5.2), each leader communicates its profit parts map to the other leader and informs its old coalition members of the negotiation results and the new leader, so that every member can compute the new map and send it to the host.

4.3 Normalization of Offer Values

If profits were hierarchically equally shared in each negotiation, and C would have joined a coalition $(A:\frac{1}{2}, B:\frac{1}{2})$, the new map would be $(A:\frac{1}{4}, B:\frac{1}{4}, C:\frac{1}{2})$.

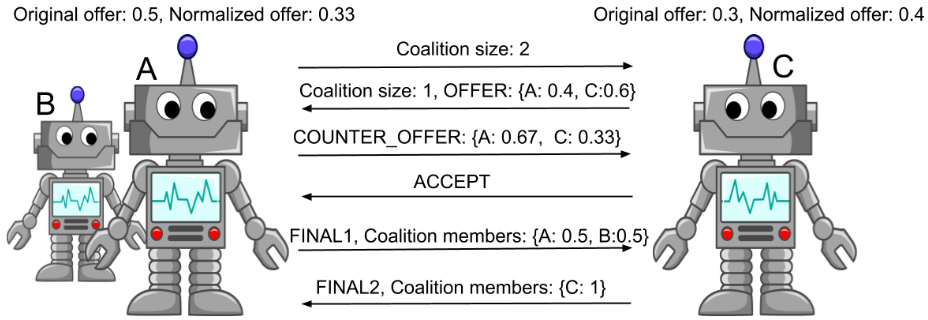


Fig. 2. Coalition Protocol: Two leaders negotiating a merge.

Eventually, even though the aggregate profit of a large coalition is larger, profit parts of early-joining agents would deteriorate. They would be incentivized to leave the coalition and start afresh, resulting in unstable coalitions.

To avoid this problem, leaders normalize the offer values. If a single agent A would have offered $0 < \alpha < 1$ to a potential single partner B, then when agents A and B lead coalitions C_A and C_B respectively, A offers B $2\alpha\frac{|C_B|}{|C_A|+|C_B|}$. As long as $0 < \alpha \leq \frac{1}{2}$, the normalized value is in $(0, 1)$. For two coalitions of size 1, the normalized value equals α. In the example above, the resulting map is $(A : \frac{1}{3}, B : \frac{1}{3}, C : \frac{1}{3})$, and agents are not incentivized to start afresh.

4.4 Leaving a Coalition

An agent who wishes to leave a coalition need only refrain from sending the map to the host. The host will exclude this agent from the coalition, normalize the other members' profit parts, and notify the other members. If the leader leaves the coalition, the coalition dissolves, and members must negotiate to create new ones.

4.5 Finding the Majority Version

The host may receive different map versions for the same coalition, e.g., when an agent wants to leave the coalition or dishonestly try to divide the profit parts differently. In this case, if there is a majority version, the host will consider it the true map. The host will remove the agents who sent a different map, normalize the map, and notify the remaining members. Otherwise (there is no majority version), the coalition is rejected, and an empty map is returned.

5 Host Policy and Agent Strategy

In view of the suggested protocol, the host and agents may use various strategies. In this paper we are interested in identifying plausible host policies which lead to our goal of stable and social-welfare-safe collaborations. To evaluate how these policies actually lead to our goals, we analyze how guests should behave under these policies to optimize their profit. This allows us to accurately model the behavior of plausible, rational guests and justify our conclusions regarding the proposed protocol.

5.1 Host Policy

In addition to collecting the agents' coalition requests and returning the resulting map, the host may enforce rules on agents' participation in coalitions.

To preserve social welfare, the host should enforce a **no-increase policy**: an agent who attempts to increase a bid quantity will be removed from the coalition. We will demonstrate how without this policy, allocation efficiency is compromised. Agents evaluate their *utility* from a possible bid quantity q as $U(q) = V(q) - bill(q)$, where $V(q)$ is their valuation and $bill(q)$ is the estimated bill they will pay. This estimation is based on previous bills. Let i denote such an agent. Since i's valuation function V_i is monotonically increasing and concave, i's participation in a coalition increases $argmax_q(U)$ and as a result—i's bid quantity. Unless i is the last guest to be allocated a resource, this will cause at least one agent to get less of the resource. The original allocation is based on truthful bids and therefore is approximately optimal (as in [27]), so any deviation from it reduces the social welfare.

The no-increase policy does not impede the system's ability to accommodate changes in the guest valuation function over time. If the load on the guest increases, it will be worthwhile to the agent to increase its bid quantity and leave the coalition.

A host may decide that only agents allocated their full bid quantity may be admitted to a coalition. We define such agents as **satisfied**. Allowing only satisfied agents in a coalition protects the host from a possible strategy of unsatisfied agents: if they can join a coalition, they can attempt to increase their compensation by pretending that an allocation causes them more harm than it actually does. They do so by bidding for a large quantity in advance, knowing they will neither win it nor pay for it but will still get compensated by other coalition members.

With these precautions, the social welfare is likely to be preserved. However, the mechanism is still vulnerable to agents leaving the coalition, increasing their resource demand, and then rejoining a coalition. Allowing only satisfied guests in coalitions reduces the potential profit of this scheme, but does not eliminate it completely.

5.2 Agent Strategy

Given the protocol we defined and the host's policy, what should the agent's strategy be?

Bill Estimation SMPSP agents are *p-truthful*: for a bid quantity q_i, and valuation function V, the best strategy for choosing a bid price is $p_i(q_i) = \frac{V(q_i+base)-V(base)}{q_i}$ [13]. So, agents need only decide on the bid quantity.

A rational agent in a coalition should weigh the consequences of increasing q_i, assuming its bill is undiscounted for the new amount, at least until rejoining a coalition. Let $D_i(q_i)$ denote agent i's estimated discount from the coalition. i estimates it will take N_1 rounds to rejoin a similar coalition, and N_2 rounds for the game to end or for i's valuation to change. Finally, let q_i^{prev} denote i's previous bid quantity. So, when considering a bid quantity q_i, i will estimate the expected value of its future bill as:

$$bill_{est}(q_i) = \begin{cases} B_i(q_i) - D_i(q_i), & \text{if } q_i \leq q_i^{prev} \\ B_i(q_i) - D_i(q_i)\frac{N_2 - N_1}{N_2}, & \text{otherwise} \end{cases} \tag{3}$$

In other words, the agent expects to lose the discount (due to the host's no-increase policy) until rejoining the coalition after N_1 rounds. However, if the agent does not increase q, then it expects to stay in the coalition and continue paying the discounted price. This means that agents will avoid changing their bid quantity unless necessary, for example when their valuation changes.

Agent i can estimate $B_i(q_i)$ and $D_i(q_i)$ on the basis of i's previous bills and discounts. Agent i can estimate the time during which its own load and valuation are expected to remain the same. When it expects its resource requirements to change, it also expects the coalition to break. In addition, i may learn the typical duration of a coalition, according to its experience with specific partners. N_1 can be estimated optimistically as $log_2(|coalition|)$, as would happen if all possible coalitions are formed in each round, or pessimistically as $|coalition| - 1$, as would happen if a single guest joins the coalition in each round. N_1 and $D_i(q_i)$ also depend on the agent's negotiation strategy.

Negotiation Strategy The negotiation strategy of an agent is a pair (OV, AT), where Offer Value $OV \in (0, 1)$ is the initial profit part offer the agent proposes, and Accept Threshold $AT \in (0, 1)$ is the minimal profit part the agent is willing to accept. A high AT and low OV will increase the agent profit ($D_i(q_i)$) in a given coalition. However, such values will discourage potential partners from forming a coalition with the agent, i.e., increase N_1. Suppose the agent has learned the others' values, e.g., from previous negotiations. Then to increase its chances of joining a coalition sooner rather than later, the agent should choose a high OV, just above the others' values, and a suitably low AT. This will maximize the number of rounds the agent spends in coalitions and the total profit over time (a formal analysis is available in the appendix). Hence, a uniform environment, where all agents have the same (OV, AT), is unstable. We focus instead on a mixed environment, where each agent has its own negotiation profile, and we conduct experiments to find the optimal values.

6 Experimental Evaluation

We experimented to find good strategies in a mixed environment, to determine the effect of coalitions on agent profits and social welfare, and to characterize the connection between coalition size and agent profits.

6.1 Experimental Setup

Experiments were run on a server with 24 cores and 16GB of RAM, running Ubuntu Linux 4.4.0-53-generic. Agents were run on virtual machines in an implementation of Ginseng [13]. The code was released as free software and is available from [9].

Our experimental environment represents a cloud machine, with 10 VMs running servers. Each server runs elastic memcached [13]—an elastic memory version of a widely used key-value cloud application. Memcached has a concave performance function. For the valuation function we used $V(performance) = c \cdot performance$. c is a constant, chosen for each guest i.i.d. from a Pareto distribution with $\alpha = 1.36$, as in [13]. The available RAM for auction was determined as one-third of the aggregate demand, to create resource pressure, under which there are payments in the SMPSP auction.

6.2 Optimal Negotiation Profile

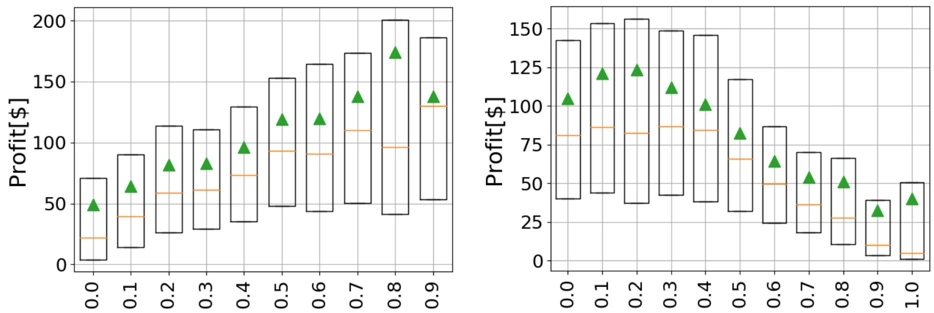
We restricted the agent strategy to choosing a constant negotiation profile and analyzed which profiles yield the most profit. Analyzing strategies which evolve over time and depend on partner identity is left for future work.

To create a mixed environment, we sampled 20 valuation sets and ran 5 experiments of 40 rounds each for each set. In each experiment OV and AT were drawn i.i.d. from a uniform distribution. $OV \sim U(0, 1)$, and $AT \sim U(0, 1-OV)$, since an agent should not decline a value above what would be obtained from offering OV. If after normalization (§4.3) $OV > 1$, then the agent would refrain from offering it. All agents were allowed to participate in coalitions. Agents' valuations were shuffled every 10 rounds so that coalitions would break (due to the no-increase policy), allowing for more negotiation opportunities.

In a mixed environment, accept thresholds matter. Too high a threshold means being too "picky". Too low a threshold means settling for deals with lower profits. Offer values matter as well. When the offer value is low, the agent keeps more of the profit, and earns more from each coalition. When it is high, so is the agent's chance of joining a coalition. In our settings, the optimal AT is around 0.8 (Fig. 3a) and the optimal OV is around 0.2 (Fig. 3b). The profit peak at $OV = 0.2$ is caused by the way ATs are drawn: over half of the ATs drawn this way are below 0.2. Therefore, if an agent's OV is higher than 0.2, most partners are likely to accept it. As OV grows, the agent relinquishes a larger part of the profit to its partner, without notably increasing the chances of acceptance while decreasing profits. Exact values depend on the profile distribution.

6.3 Social Welfare and Guest Profit

How are the social welfare and profit affected by negotiations? Given our results in §6.2, we chose a symmetric profile that is likely for rational agents: $OV = 0.2$ and AT just below it at 0.15, so that coalitions can form. With this profile we ran three experiment scenarios: coalitions are allowed for satisfied agents, coalitions



(a) Accept Threshold (b) Offer Value

Fig. 3. Profit from coalitions as a function of negotiation parameters in a mixed environment. Mean values are shown as triangles. The boxes represent the first, second and third quartiles.

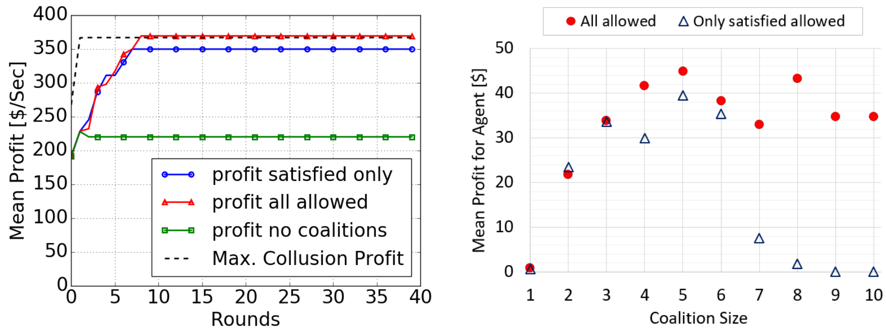
are allowed for all agents, or coalitions are prohibited. Each experiment ran for 40 rounds. Valuations for the guests are as in §6.2, but they were not shuffled.

In each experiment, we calculated the social welfare and the mean profit for guests in each round. The profit is the difference between the valuation of the resource and the actual (occasionally discounted) amount paid. While the social welfare remains the same (i.e., the allocation of goods remains efficient), the mean profit increases by up to 67% (60% on average), as shown in Fig. 4a. When coalitions are allowed for all agents, the mean profit reaches the profit from the optimal collusion scheme, where guests share their entire private valuations and bid for the same quantities they would get from the host, so their bill is zero.

6.4 Optimal Coalition Size

What is the optimal coalition size for agents? We ran two experiment batches, one with only satisfied guests and one with all guests. In each batch, 5 different valuation sets were drawn. Each set was run 10 times.

Although a single coalition of all agents has the largest aggregate profit, it does not necessarily optimize the mean profit of a single member. In fact, the opposite is true: some agents, when joining the coalition, do not increase the total coalition profit but share it: their mean profit does not monotonically increase with the coalition size, (see Fig. 4b). This result holds whether or not unsatisfied agents are allowed. It means that a single coalition of all agents is not in the agents' best interest, and so it is unstable. The instability can be resolved using a Shapley value based division, where each agent's profit depends on its contribution.



(a) Coalition effect on profit (b) Coalition size effect on profit

Fig. 4. Fig. 4a shows the mean profit for guests in each round, for different host coalition support policies. Fig. 4b shows the mean profit from coalitions per agent in a single round as a function of the coalition size.

7 Conclusion and Future Work

As cloud computing advances, we expect more clouds to shift to economic mechanisms such as auctions for selling cloud resources. As they do so, the importance of understanding and handling collusion between auction participants will grow. We presented and implemented a negotiation mechanism for coalition formation in a VCG-like auction, which is suitable for auctioning computing resources. It maintains the auction's social welfare while reducing the participants' costs. Although coalition participants pay lower bills, they nonetheless refrain from unnecessarily increasing their resource demands. In a mixed environment, profit will be maximized for an agent willing to accept a profit share of at least 45% and who offers partners 40%. Nor do large coalitions necessarily benefit their members. We conjecture that negotiation strategies that change over time and with different partners may increase the agents' profit. Exploring different division schemes such as Shapley [39]-based division is left for future work.

VCG auctions can also be used for multi-resource allocation using the same exclusion-compensation principle as for a single resource. Therefore, given a VCG multi-resource auction implementation, the anti-collusion coalition mechanism can be applied to it as well.

8 Acknowledgments

This work was partially funded by the Amnon Pazi memorial research foundation. We thank D. Parkes for fruitful discussions. We thank A. Bousso, Y. Lev, A. Ohayon, and S. Levenzon Kuninin for their contributions to the software implementation.

References

1. Amazon EC2 pricing. Website. https://aws.amazon.com/ec2/pricing/

2. AWS Lambda. Website. https://aws.amazon.com/lambda/
3. Azure functions pricing. Website. https://azure.microsoft.com/en-us/pricing/details/functions/
4. Azure pricing. Website. https://azure.microsoft.com/en-us/pricing/
5. CloudSigma price schedules: Burst pricing. Website. URL http://cloudsigma-docs.readthedocs.io/en/2.14/billing.html#burst-levels
6. GCP pricing — google cloud platform. Website. https://cloud.google.com/pricing/
7. Spot instances - product introduction— alibaba cloud documentation center. Website. https://www.alibabacloud.com/help/doc-detail/52088.htm
8. Spot marketing pricing - discount packet bare metal servers. Website. https://www.packet.net/bare-metal/deploy/spot/
9. Agmon, S., Kuninin, S., Bousso, A., Lavi, N.: Raas negotiations. https://bitbucket.org/shunit/negotiations (2018)
10. Agmon Ben-Yehuda, O., Ben-Yehuda, M., Schuster, A., Tsafrir, D.: The Resource-as-a-Service (RaaS) cloud. In: Proceedings of the 4th USENIX Conference on Hot Topics in Cloud Computing, p. 12. USENIX Association (2012)
11. Agmon Ben-Yehuda, O., Ben-Yehuda, M., Schuster, A., Tsafrir, D.: Deconstructing Amazon EC2 spot instance pricing. ACM Trans. Econ. Comput. **1**(3) (2013)
12. Agmon Ben-Yehuda, O., Ben-Yehuda, M., Schuster, A., Tsafrir, D.: The rise of raas: the resource-as-a-service cloud. Communications of the ACM **57**(7), 76–84 (2014)
13. Agmon Ben-Yehuda, O., Posener, E., Ben-Yehuda, M., Schuster, A., Mu'alem, A.: Ginseng: Market-driven memory allocation. SIGPLAN Not. **49**(7), 41–52 (2014)
14. Agmon Ben-Yehuda, O., Schuster, A., Sharov, A., Silberstein, M., Iosup, A.: Expert: Pareto-efficient task replication on grids and a cloud. In: 26th IEEE International Parallel and Distributed Processing Symposium, IPDPS, pp. 167–178 (2012)
15. Aoyagi, M.: Bid rotation and collusion in repeated auctions. Journal of Economic Theory **112**(1), 79–105 (2003)
16. Aoyagi, M.: Efficient collusion in repeated auctions with communication. Journal of Economic Theory **134**(1), 61–92 (2007)
17. Bachrach, Y.: Honor among thieves: collusion in multi-unit auctions. In: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: Volume 1, pp. 617–624. International Foundation for Autonomous Agents and Multiagent Systems (2010)
18. Blume, A., Heidhues, P.: Modeling tacit collusion in auctions. Journal of Institutional and Theoretical Economics JITE **164**(1), 163–184 (2008)
19. Chatterjee, K., Mitra, M., Mukherjee, C.: Bidding rings: A bargaining approach. Games and Economic Behavior **103**, 67–82 (2017)
20. Clarke, E.H.: Multipart pricing of public goods. Public Choice **11**(1), 17–33 (1971)
21. Eső, P., Schummer, J.: Bribing and signaling in second price auctions. Games and Economic Behavior **47**(2), 299–324 (2004)
22. Funaro, L., Agmon Ben-Yehuda, O., Schuster, A.: Ginseng: Market-driven LLC allocation. In: 2016 USENIX Annual Technical Conference, pp. 295–308. USENIX Association, Berkeley, CA, USA (2016)
23. Graham, D.A., Marshall, R.C.: Collusive bidder behavior at single-object second-price and English auctions. Journal of Political Economy **95**(6), 1217–1239 (1987)
24. Groves, T.: Incentives in teams. Econometrica **41**(4), 617–631 (1973)
25. Johnson, P., Robert, J., et al.: Collusion in a model of repeated auctions. Université de Montréal, Centre de recherche et développement en économique (1999)

26. Kraus, S., Shehory, O., Taase, G.: The advantages of compromising in coalition formation with incomplete information. In: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 2, pp. 588–595. IEEE Computer Society (2004)
27. Lazar, A., Semret, N.: Design and analysis of the progressive second price auction for network bandwidth sharing. Telecommunication Systems—Special issue on Network Economics **20**, 255–263 (1999)
28. Mailath, G.J., Zemsky, P.: Collusion in second price auctions with heterogeneous bidders. Games and Economic Behavior **3**(4), 467–486 (1991)
29. Maillé, P., Tuffin, B.: Multi-bid auctions for bandwidth allocation in communication networks. In: IEEE INFOCOM (2004)
30. Marshall, R.C., Marx, L.M.: Bidder collusion. Journal of Economic Theory **133**(1), 374–402 (2007)
31. McAfee, R.P., McMillan, J.: Bidding rings. The American Economic Review pp. 579–599 (1992)
32. Metz, C.: Facebook doesn't make as much money as it could - on purpose. Website (2015). https://www.wired.com/2015/09/facebook-doesnt-make-much-money-couldon-purpose
33. Movsowitz, D., Agmon Ben-Yehuda, O., Schuster, A.: Attacks in the resource-as-a-service (RaaS) cloud context. In: N. Bjørner, S. Prasad, L. Parida (eds.) Distributed Computing and Internet Technology, pp. 10–18. Springer International Publishing (2016)
34. Movsowitz, D., Funaro, L., Agmon, S., Agmon Ben-Yehuda, O., Dunkelman, O.: Why are repeated auctions in RaaS clouds risky? In: Economics of Grids, Clouds, Systems, and Services - 15th International Conference, GECON 2018, Pisa, Italy, September 18-20, 2018, Proceedings. Springer (2018)
35. Rachmilevitch, S.: Endogenous bid rotation in repeated auctions. Journal of Economic Theory **148**(4), 1714–1725 (2013)
36. Rachmilevitch, S.: Bribing in second-price auctions. Games and Economic Behavior **92**, 191–205 (2015)
37. Ristenpart, T., Tromer, E., Shacham, H., Savage, S.: Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In: Proceedings of the 16th ACM conference on Computer and communications security, pp. 199–212. ACM (2009)
38. Seres, G.: Auction cartels and the absence of efficient communication. International Journal of Industrial Organization **52**, 282–306 (2017)
39. Shapley, L.: Stochastic games. Proceedings of the National Academy of Science of the USA **39**(10), 1095–1100 (1953)
40. Skrzypacz, A., Hopenhayn, H.: Tacit collusion in repeated auctions. Journal of Economic Theory **114**(1), 153–169 (2004)
41. von Ungern-Sternberg, T.: Cartel stability in sealed bid second price auctions. The Journal of Industrial Economics pp. 351–358 (1988)
42. Varian, H.R., Harris, C.: The VCG auction in theory and practice. American Economic Review **104**(5), 442–45 (2014)
43. Vickrey, W.: Counterspeculation, auctions, and competitive sealed tenders. J. of Finance **16**(1) (1961)
44. Yu, D., Mai, L., Arianfar, S., Fonseca, R., Krieger, O., Oran, D.: Towards a network marketplace in a cloud. In: HotCloud (2016)

9 Appendix

In this appendix we provide a mathematical model for the expected profit of an agent as a function of its offer value, and show that this profit depends greatly on the specific strategies used by other agents in the environment. We use the same notations as in section 5.2. N_1 denotes the number of rounds required to create a coalition of all N agents in the system, and N_2 denotes the number of rounds the coalition is expected to hold until it breaks (as a result of a change in guest load or a change in the guest population on the machine). We analyze a cycle of $N_1 + N_2$ rounds. Assume guest i has $AT = 1 - OV$. Then in each coalition, the fraction f of the coalition discount D that i would get is at least $1 - OV$. In one cycle, the expected profit from coalitions per round for a guest i is

$$p_i(f) = \frac{N_2}{N_1 + N_2} f D. \tag{4}$$

N_1 is a monotonically increasing function of f: the higher the AT, the harder it is for i to join a coalition. Note that N_1 is also monotonically increasing in N (the number of agents), since the other agents could join the coalition before i. However, we currently assume the number of guests is approximately constant, i.e., changes significantly more slowly than the guest load. Then we can write:

$$N_1 \sim h(f) \tag{5}$$

where h is an unknown monotonically increasing functions. h represents the specific agent strategy for response on offers, for example, how the agent chooses among multiple acceptable offers. Assume N is constant. Using Tailor approximation of the second order, we can write:

$$N_1(f) = C + Af + Bf^2 + O(f^3) \tag{6}$$

Substituting this into the profit expression, we have:

$$p_i(f) = \frac{N_2}{C + Af + Bf^2 + N_2} f D \tag{7}$$

The agent would like to maximize this function by changing f, or equivalently, minimize the following expression:

$$\begin{aligned} \frac{D}{p_i(f)} &= \frac{C + Af + Bf^2 + N_2}{N_2 f} = \\ &= \left(\frac{C}{N_2} + 1\right)\frac{1}{f} + \frac{A}{N_2} + \frac{B}{N_2} f \end{aligned} \tag{8}$$

The first term is monotonically decreasing in f. The second is constant, while the third is monotonically increasing with f. Therefore, the optimal f depends on the h function, that is, the specific agent strategy. This means that for different environments, depending on the specific implementation of the agent, the optimal f values are different, and as a result, so are the optimal OV, AT values.