Applications of Mechanism Design to Cloud Computing Upgrades

Will Sherwood

May 8, 2020

1 Introduction

Suppose N companies are using a cloud provider service. That is, each company can send a computation request to the nearest data center in the cloud network, that data center will process and compute the request, and then the output will be sent back to the company. Cloud computing and infrastructure as a service has been a large development in technology in the last ten years, with a current market value of over 250 billion dollars.

In practice, the cloud network providers may want to build more infrastructure to improve their services. Although some companies may want to pay for upgrades, it may not make sense to charge each company the same amount. For example, suppose there is only one data center in the cloud service located in Southern California, and there are two companies using the cloud provider – one in Northern California, and one in New Mexico. Then, the company located in Northern California stands to benefit much more from the cloud provider building a data center in San Francisco than does the company in New Mexico, and therefore it should be charged more to balance this. Suppose that the cloud provider declared that it was going to build a data center in Northern California, and that each company using the cloud provider should pay their fair share based on how much utility they would each receive from the upgrades. However, because the cloud provider does not necessarily have utility insights into its clients, each company could misrepresent their utility to the cloud provider, and pay prices other than their fair share.

How can we solve this problem? Is there a protocol that the cloud provider could broadcast, so that truth-telling can be proven optimal in polynomial time in the number of companies and outcomes? It turns out that the answer is yes. We will present an efficiently verifiable protocol where the cloud provider can poll the tenants for information, charge them individualized fees based on their collective polling responses, and implement a decision. The key insight here is that the system can be set up in a way so that each company is incentivized to act truthfully – that is, lying is never an economically rational decision. Additionally, the tenants learn negligible information about the other corporations.

2 Mechanism Design

Mechanism design, a well studied class of problems in algorithmic game theory, can be used to tackle this problem. Formally, a mechanism on r types and n outcomes with expected utility $T : [r] \times [n] \to \mathbb{R}$ is a game consisting of a map $g : [r] \to [n]$ and a payment $p \in \mathbb{R}^n$. The player in the game has a true type $t \in [r]$, is given the pair (g, p), and declares a type $t' \in [r]$. The player receives $T(t, g(t')) - p_{g(t')}$ utility, and the game terminates.

To move towards a solution of the proposed problem, we need to figure out a way given the utility space T, to come up with a mechanism and payment pair (g, P) so that the player reveals their true type t. Notice that a player may be incentivized to lie if we do not carefully set up the game. Lying may be an optimal strategy for this game if we overcharge them for their true type. That is, assuming the player is trying to maximize their expected utility from playing the game, they may be better off declaring a different type $t' \neq t$ if $T(t, g(t')) - p_{g(t')} > T(t, g(t)) - p_{g(t)}$.

Natural questions to ask in this context are when incentive compatible mechanisms for a type space exist, and whether or not they are efficiently computable. The following two lemmas will answer those questions in the positive.

Lemma 1. Every type space T and price vector $p \in \mathbb{R}^n$ has an incentive compatible mechanism g, and one can be computed in time $O(rn)$.

Proof. For each type $t \in [r]$, consider the vector $T(t, -) - p$, whose ith coordinate is the overall expected utility a player with true type t obtains under outcome i. Now, simply take the coordinate $k \in [n]$ for which this is maximized, and call that $g(t)$. Since every player is getting their best outcome (ordered by expected utility) under the map g, declaring their true type is always an optimal strategy. Hence (g, p) is an IC mechanism pair. The time complexity is optimal, since at least, any algorithm solving this problem must look at every entry of T, which already takes $O(rn)$ time.

Lemma 2. Given a type space T and mechanism $g : [r] \to [n]$, if there exists a $p \in \mathbb{R}^n$ for which (g, p) is an IC mechanism pair, p can be computed efficiently.

Proof. Let $p_1, \cdots, p_n$ be formal variables. Then, the condition for IC gives us that for each pair of types $t \neq t' \in [r]$, $T(t, g(t')) - p_{g(t')} \leq T(t, g(t)) - p_{g(t)}$. But this is a series of linear inequalities in the formal variables p_i. Therefore, the existence of a price p for which (g, p) is an IC mechanism pair can be determined and displayed efficiently using a linear program of size $O(rn)$.

Full classification of incentive compatible mechanisms can be found at [1]. However, the use of these two lemmas will allow us to solve the data center problem as described in the introduction. More generally, I hope that this demonstrates that even in complicated theories, you can often get the results you need to solve the problems that are important to you with first principles.

3 Revenue Maximization

Given an incentive compatible mechanism (g, p) and a probability distribution D over the type space, the expected revenue can be simply computed. If outcome i costs S_i in expectation for the cloud to implement, the cloud's expected revenue can be computed as

$$\mathbb{E}[\text{Cloud Revenue}(g, p)] = \sum_t Pr[t \sim D](p_{g(t)} - S_{g(t)}).$$

For a given cloud tenant with type t, they can compute their own expected revenue simply as well:

$$\mathbb{E}[\text{Tenant Revenue}_t(g, p)] = T(t, g(t)) - p_{g(t)}.$$

Notice that these formulae are operating under the assumption that the company's revenue maximizing optimal strategy is to declare their true type to the mechanism. This is exactly the condition that the

mechanism is incentive compatible. Now, the problem of finding a mechanism to solve the data center problem is reduced to searching over the space of incentive compatible mechanisms to find one with maximal expected revenue for the cloud company.

Unfortunately, in general, finding a mechanism pair for a given type space which approximately maximizes expected revenue that has an approximation ratio strictly greater than 1/2 is NP-Hard, even if the type space is well structured (specifically supermodular, see [2]). However, both the algorithms for generating mechanisms from prices and the algorithm for generating prices from mechanisms are efficient. Therefore, we propose the following technique to reduce some of the expression in the outcome space: Simply, discretize the type space as fine as needed. Then, for the outcomes, pick a constant k much smaller than the total number of outcomes (which are themselves often discretized). Then, make the set of outcomes to be k *probability distributions* over the outcome space (rather than points of the outcome space), and recompute the expected utilities according to the outcomes. The smaller the number of probability distributions, the more expression is given up in the outcome space. However, this greatly reduces the search space for revenue maximizing mechanisms, and works well with many real world applications of mechanism design. In general, approximating optimal revenue to mechanisms is a deep field, and one might seek to utilize other techniques in different situations [3].

After this reduction, the problem of searching for a revenue maximizing mechanism becomes efficient. That is, there are only polynomially many incentive compatible mechanisms g when there are polynomially many types but only a constant number of outcomes. (This can be seen by relating incentive compatible mechanisms with tropical hyperplane arrangements, but is out of the scope of this thesis [1]). In the following section, we will describe exactly how to set up the problem described in the introduction as a mechanism, employ the output space compression technique, and find an expected revenue maximizing incentive compatible mechanism to solve the problem.

4 Metric Induced Mechanism

Latency-sensitive applications running on cloud infrastructure stand to gain the most from adding new data centers near their location. [**Kwon**] In this section, we will demonstrate the power of mechanism design to solve the issue proposed in the introduction. More specifically, we derive a utility function from a metric in the scenario of network latencies, and present an incentive compatible, positive expected revenue mechanism to upgrade and fund a new data center. All of the code for this example was made to be easily extendable to generate IC, positive expected revenue mechanisms for arbitrary type spaces, and is included in the appendix.

Let $E = [-1, 1]^2$ be the unit square with the distance between two points representing the number of seconds of latency between data centers built at that point. Set the cloud to be $C = \{(-1/2, -1/2), (1/3, -1/3)\} \subset E$ to be two data centers located at those respective points.

Now, as set up in the introduction, suppose the cloud provider decides to build a new data center, using one of the probability density functions

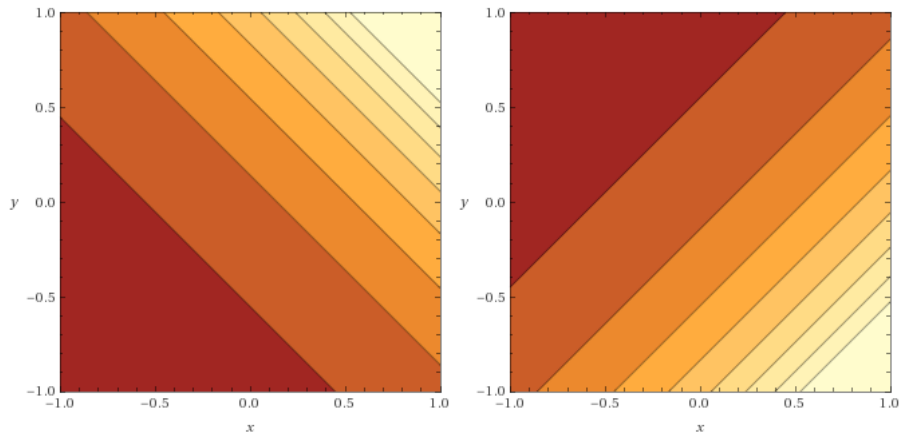
$$D_\sigma(x, y) = \frac{\exp(\sigma^1 x + \sigma^2 y)}{e^2 + 1/e^2 - 2}, \qquad \sigma^i = \pm 1$$

which, after fixing σ^1 and σ^2, map corners of the square with increasing probability (see Figure 1).

Given ϵ, let E_ϵ be the discretized unit square

$$E_\epsilon = \{x - (1, 1), x \in [2/\epsilon]^2\}$$

Figure 1: $\sigma = \{1, 1\}, \{1, -1\}$



and simply choose the possible valuations of 1ms latency decreases as just one or two dollars. Then, for a company located at point $(x, y) \in E_\epsilon$ valuing each 1ms latency decrease by $\$k$, their expected utility under distribution D_σ is

$$\mathbb{E}[Utility((x, y), k, \sigma)] = \frac{k}{\text{Vol}E} \int_{E \cap B((x,y),\xi)} D_\sigma(t)(\xi - ||t - (x, y)||)dt$$

where $B(x, \delta)$ is the ball of radius δ centered around x, and $\xi = \text{min-lat}_C(x, y) = \min_{c \in C} ||(x, y) - c||$.

Notice that companies located nearest the cloud stand to gain the least here, since they already have very low latency. Since the data center points are located at $(-1/2, -1/2)$ and $(1/3, -1/3)$, companies located near the $(-1, 1)$ corner stand improve their latency to the cloud under their best plan over other companies (150ms). A company located near the $(1, 1)$ corner also stands to improve their latency to the cloud significantly, by around 100ms. Recall that the second company's expected utility could still be higher than the first if their valuation of latency to cloud is twice that of the first.

For notational convenience, let $[n]$ denote the set $\{0, 1, \cdots, n\}$. Set $\epsilon = 1/10$, $P = \{1, 2\}$, and let E_ϵ be the discretized unit square,

$$E_\epsilon = \{x/10 - 1, x \in [20] \times [20]\}$$

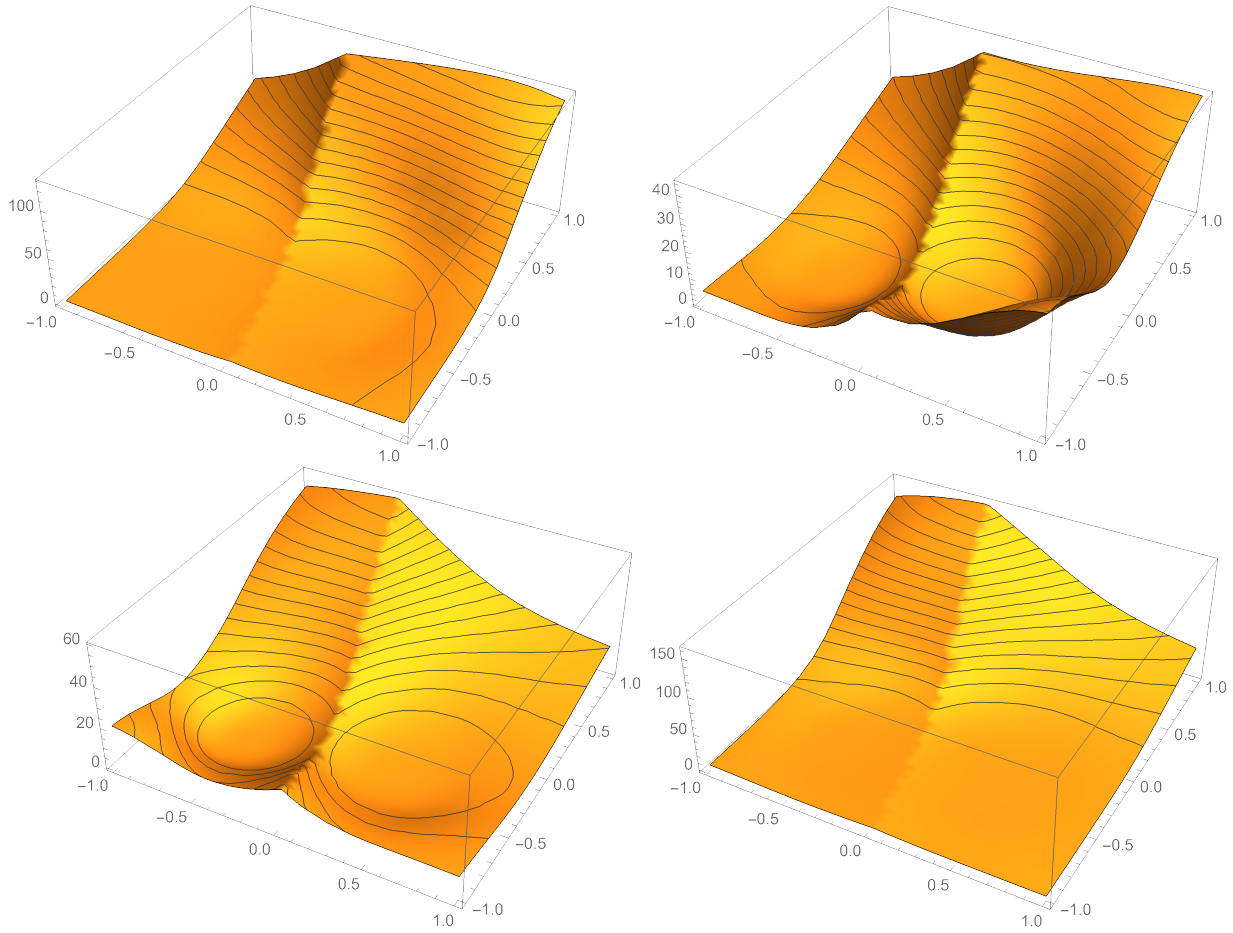
Now, we can compute incentive compatible mechanisms for this discretized type space using the provided code and compute the expected revenue. Let $S \in \mathbb{R}^n$ denote the expected cost of building a data center according to each outcome distribution. For notational convenience, denote the utility that a company with type t gets under outcome $g(t)$ as $U(t, g(t))$. Then, under mechanism (g, p) and priors D on the type space, the expected revenue for the cloud can be calculated as

$$\mathbb{E}[\text{Cloud Revenue}(g, p)] = \sum_t Pr[t \sim T](p_{g(t)} - S_{g(t)})$$

and each company, with type t, will have expected revenue

$$\mathbb{E}[\text{Company Revenue}(g, p, t)] = T(t, g(t)) - p_{g(t)}$$

Figure 2: Utility landscapes for $k = 1$, $\sigma = \{1, 1\}, \{1, -1\}, \{-1, 1\}, \{-1, -1\}$, starting in the upper left corner and going clockwise.



For every company to want to participate in the mechanism, it is a requirement that they are expected to gain utility from it. Otherwise, they may not want to elect to have any datacenters built at all. Therefore, p must satisfy

$$\text{For all } t, \qquad p_{g(t)} \leq \mathrm{U}(t, g(t)).$$

That is, $p_i \leq \min_{\tau \in g^{-1}(i)} \mathrm{U}(\tau, i)$

To restrict our focus to mechanisms satisfying this property, we will need the following lemma:

Lemma 3: If $p \in \mathbb{R}^n$ is an incentive compatible payment vector for $g : E_\epsilon \to \{-1, 1\}^2$, and $\lambda \in \mathbb{R}$ is any constant, then $p' = \{p_i + \lambda, p_i \in p\}$ is also an incentive compatible payment vector for g. That is, incentive compatible payments are closed under addition.

Proof: For p to be an incentive compatible payment vector for g, that means that for every type $t \in E_\epsilon$, and any outcome o which is not the outcome for t under g, the expected company revenue must not exceed the expected company revenue if a different outcome were to happen. That is,

$$\text{For all types } t \in E_\epsilon, \text{ and } o \neq g(t),\ U(t, g(t)) - p(g(t)) \geq U(t, o) - p(o)$$

But then, replacing $p_i \mapsto p_i + \lambda$ in the above immediately gives that p' is an incentive compatible payment for g.

Therefore, in order for the cloud provider to maximize its own expected revenue according to their priors

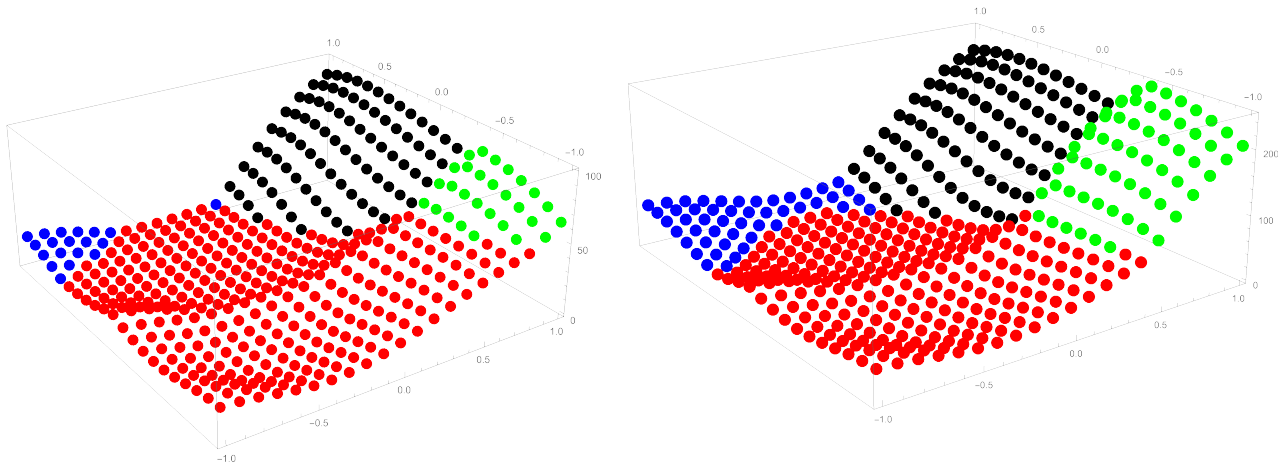
over the type space, while still ensuring that their customers will participate in the game, they should scale any p by the value

$$p \mapsto p - \min_i \min_{\tau \in g^{-1}(i)} U(\tau, i) - p_{g(\tau)}.$$

In this case, notice that for any $\epsilon > 0$, some point of E_ϵ will be ϵ away from a data center by definition. In particular, this means that, regardless of the outcome, at least one of the prices charged to the participants will be tending towards a value which is at most zero. (A negative price here simply denotes the cloud provider must *pay* the participants to build a datacenter). Since building datacenters cost money, but some outcome will inevitably be free, it seems like the task to find an incentive compatible mechanism with positive expected profit for both the cloud provider and tenants would be difficult. This is where the power of mechanism design begins to speak for itself – Below are several mechanisms under different conditions all with positive expected profit for both the cloud provider and tenants.

4.1 Mechanism 1a.

For the two example mechanisms described here, suppose that the distribution of company locations and valuations is uniform. Say $S = \{15.25, 20, 15.5, 5\}$ are the expected prices to implement each respective outcome. In particular, we are not assuming that each distribution of data center location costs the same to draw and build from, in expectation. The code included in the appendix of this project was used to generate an incentive-compatible mechanism with approximately optimal revenue for the cloud provider, and positive expected utility for any company regardless of location.



Optimal expected revenue mechanism when $S = \{15.25, 20, 15.5, 5\}$.
Red $\mapsto \{-1, -1\}$, Green $\mapsto \{-1, 1\}$, Blue $\mapsto \{1, -1\}$, Black $\mapsto \{1, 1\}$.

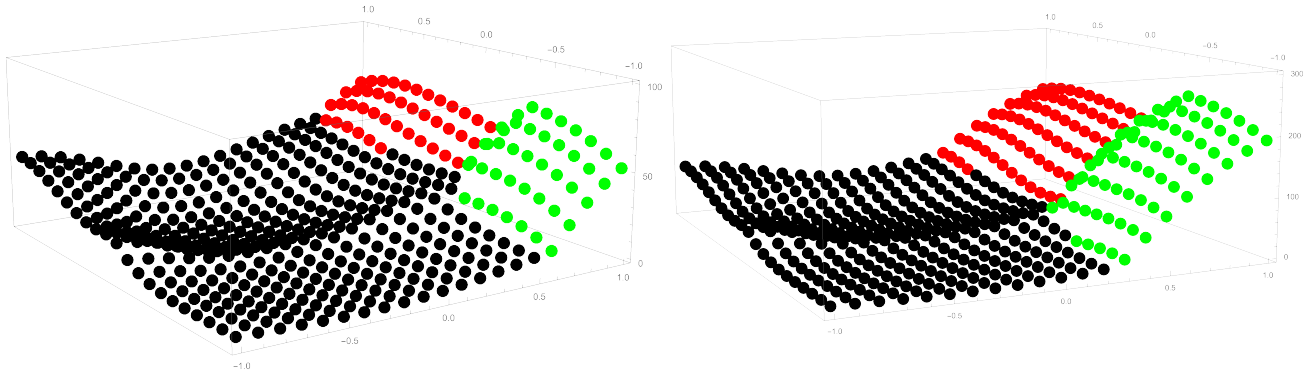
The figure above is a representation of the mechanisms. On the left, the points correspond with types of the discretized unit square and a latency valuation of 1 ms $\iff$ \$1. On the right, it shows when the valuation is \$2. Each color represents the outcome distribution on new data center locations if a company declares their type to be centered at the point. The z-axis shows the expected utility that a company at the given location will gain under rational (optimal) play of the mechanism, after payments. The prices charged for each outcome are $\{0, 75.7, 20.6, 34.8\}$, respectively.

The program supplied in the Appendix will find revenue maximizing mechanisms for any probability distributions on types. For example, suppose that the cloud provider believed that a company was more

likely to exist in a particular location if their utility could be maximized for the mechanism. Specifically, weight the probability of a company having a particular type by their utility. In this case, with the same S vector, the revenue maximizing mechanism has prices $\{0, 74.5, 58.7, 69.6\}$. Comparing with the above, this situation expects to have a much higher expected revenue ($45 in this case) than without weighting by utility. The mechanism here loads off most types to the cheapest outcomes, and then charges the companies in a position to make the most utility a hefty price for it. Overall, the third and fourth outcome are both double that of the revenue maximizing mechanism with respect to the uniform distribution on types.

4.2 Mechanism 2a.

To give another example, suppose we made the distribution $\{-1, -1\}$ cost $\$10,000$ in expectation to implement, and the others free. Clearly, any revenue maximizing mechanism would avoid mapping to $\{-1, 1\}$ at all costs, since it would massively drive down the profit gained from the cloud provider. However, the class of incentive compatible mechanisms is able to account for this, and the code provided in the appendix once again finds an incentive compatible mechanism with positive expected revenue for all parties involved.



Optimal expected revenue mechanism when $S = \{10000, 0, 0, 0\}$.
Blue $\mapsto \{-1, -1\}$, Green $\mapsto \{-1, 1\}$, Black $\mapsto \{1, -1\}$, Red $\mapsto \{1, 1\}$.

Notice that the mechanism is able to completely avoid mapping to 0, while remaining incentive compatible. The prices charged for each outcome are $\{\infty, 80.99, 0, 60.60\}$.

5 Addons

5.1 Cryptographically Secure Mechanisms

In practice, it is difficult to publicly estimate the utility an agent may receive if agents are competing against each other. As in the running example, imagine a mechanism consisting of several companies which will reveal their location to a cloud provider in hopes that they will build new data centers to decrease the company's latency to the cloud. Although it is possible to compute the expected latency decrease gained by a given company under the mechanism and determine that the game is incentive compatible, computing the expected utility is not so clear. Even if the company can put a price to latency decreases, how do they factor in that they have revealed some information about their company publicly by revealing their true type to the mechanism. In fact, it may very well be the case that a mechanism is not incentive compatible, even if it seemingly is, due to this factor. (For example, one might imagine

that if a company knew the function for which another company valued decreases in latency, the former company may be able to sell this information to an internet provider for extra utility). Therefore, it would be very beneficial to be able to execute the mechanism with the following guarantees:

1. Each company should learn negligible information about the other company's types under standard cryptographic assumptions.

2. The inputs to the mechanisms (company responses) should be publicly verifiably not counterfeit, and the resulting outcome was computed correctly.

3. The output of the mechanism can be correctly computed, even if any party acted adversarially.

4. The mechanism should be efficiently provably incentive compatible.

The only cryptographic assumption we will need is that of the the quadratic residuosity problem. That is, given integers x, N (with N having two prime factors), does there exist a t such that $x = t^2 \pmod{N}$? This problem is believed to be computationally difficult, and an efficient algorithm for this would also give an efficient algorithm to decide whether any integer N can be decomposed as a product of two or three primes.

The actual cryptographic scheme is a standard application of a cryptographic counter described in [4]. The participants can use a single round protocol and send a "vote" for their type in the mechanism. The resulting outcome can be verified by any third party, and each company is charged based on the rules of the mechanism. All in all, this would allow the mechanism solving the data center problem to be cryptographically secure against the practical points made above.

6 Conclusion

We introduced a problem of placing new data centers to decrease some tenant's latencies to the cloud by differing amounts. We were able to utilize a class of algorithms from game theory to construct fair solutions to this problem. After compressing the polynomially sized possible data center build locations to a constant sized set of probability distributions over possible data center build locations, we give an algorithm for efficiently computing an optimal solution to the problem. More specifically, we give an example of a cloud with two data centers placed on the unit square, and show the revenue maximizing mechanisms. As a practical concern, we then show how to secure the mechanism using existing cryptographic primitives.

7 Acknowledgments

I would like to thank Professor Ngoc Tran for her support as an advisor throughout this project in spite of the extenuating circumstances brought on by COVID-19. Her 'virtual' office door was always open to answer my whirlwind of questions and provide feedback. It was truly an honor to take her course and to have her as my advisor. I would also like to thank my second and third readers, Professors Etienne Vouga and Professor Gordon Novak for providing their feedback on my thesis and presentation. Also, Calvin Lin, thank you so much for helping to get me on the right track with this project – this would not have been possible without your help. Thank you, Sam Gunn, for being an amazing friend and inspiration throughout my undergraduate career. Lastly, I would like to thank my parents for housing me during the pandemic, and my girlfriend Linh for the emotional support throughout this project.

References

[1] Robert Alexander Crowell and Ngoc Mai Tran. *Tropical Geometry and Mechanism Design*. 2016. arXiv: 1606.04880 [cs.GT].

[2] Uriel Feige. "On Maximizing Welfare When Utility Functions Are Subadditive". In: *SIAM Journal on Computing* 39.1 (2009), pp. 122–142. DOI: 10.1137/070680977. eprint: https://doi.org/10.1137/070680977. URL: https://doi.org/10.1137/070680977.

[3] Jason D. Hartline. "Approximation in Mechanism Design". In: *The American Economic Review* 102.3 (2012), pp. 330–336. ISSN: 00028282. URL: http://www.jstor.org/stable/23245552.

[4] Jonathan Katz, Steven Myers, and Rafail Ostrovsky. "Cryptographic counters and applications to electronic voting". In: *In Advances in Cryptology—EUROCRYPT 01*. 2001, pp. 78–92.