Risk Aware Cloud Broker System

Faculty Mentor: Dr. Ganesh Iyer
Team: Aman Arora, Pulkit Agarwal, Vimal Chaudhary

I. Introduction

A Cloud Broker System

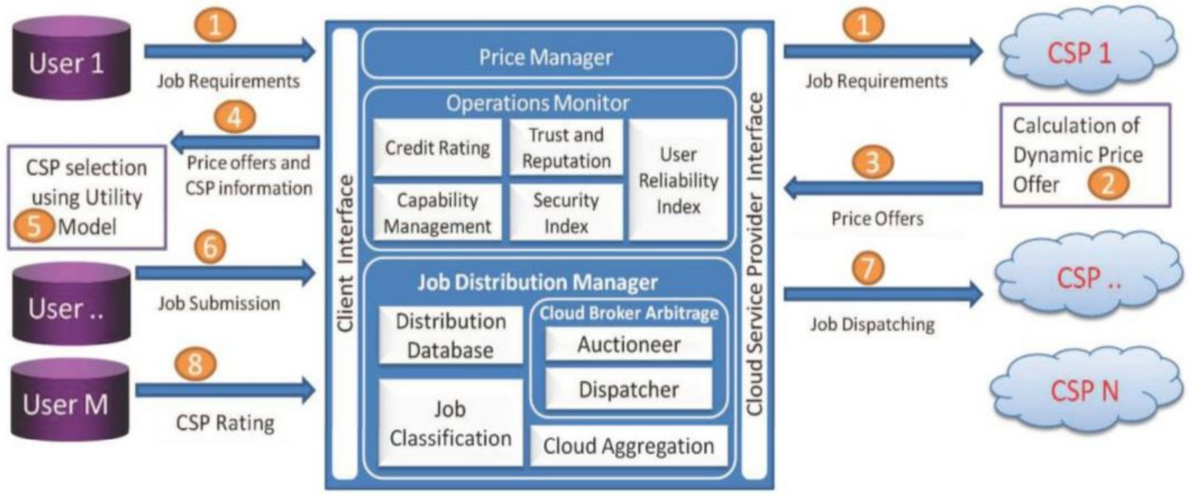
A cloud broker is a third-party individual or business that acts as an intermediary between the purchaser of a cloud computing service and the sellers of that service. In general, a broker is someone who acts as an intermediary between two or more parties during negotiations.

The broker's role may simply be to save the purchaser time by researching services from different vendors and providing the customer with information about how to use cloud computing to support business goals. In such a scenario, the broker works with the customer to understand work processes, provisioning needs, budgeting and data management requirements. After the research has been completed, the broker presents the customer with a short list of recommended cloud providers and the customer contacts the vendor(s) of choice to arrange service.

How the system works with a Broker?
Broker acts as an intermediary between Users and Cloud Service Providers (CSPs). It receives job requests from users and forwards them to CSPs. CSPs use various data from Broker for example: their market reputation, User’s reliability index, Popularity of resources in the market, etc to compute the prices which they may change dynamically over time and submits their price offerings to Broker. Broker forwards them to users, who now computes their Utility function based on data obtained from Broker, for

example: the job ratings given by other users in the past, etc. Based on the utility function users deploy, they give the job to the CSP which maximises their utility.

Thus, in this manner Broker aids the services between Users and CSPs.



Risk-based Cloud Broker Arbitrage Mechanism

- Users and Cloud Service Providers (CSPs) are connected using a Broker.
- Users submit job request to Broker who forwards this to all the connected CSPs.
- CSPs send back the price offer to the user through Broker.
- User decides offer based on his ‘utility function’.
- Utility function is based on the risk function, understood from VNM-UT.
- User tries to maximize his utility.
- CSPs update prices with time. (Dynamic pricing)
- Users give Job Rating (JR) to CSP after the job is done.

In Risk-based Cloud Broker System the utility function is modelled using the principles of Von Neumann-Morgenstern utility theorem (VNM-UT) which can effectively handle user’s risk bias towards the selection of appropriate CSPs based on their trust and reputation values.

Project Aim

Building up a mathematical modeling (using Game theory) of risk aware Cloud Brokers who to connect various Cloud Service Providers (CSPs) and users through a Broker-mediated system which will help users in making a better and more profitable decisions and will also help CSPs to adjust their behavior in order to earn maximum profit.

Motivation

There are several key challenges for both users and providers to establish such kind of service:

Users Perspective:

- So many CSPs in market which one to chose. With the variety of services offered by several CSPs, users may find it difficult to choose the right provider which matches their requirements. Without Broker, there is no uniform centralised platform which provides information about the capabilities of all the CSPs.
- Security and Privacy issues: As several users may share the same physical infrastructure in a virtualized manner simultaneously, users are often concerned about the security and privacy of their data in the Cloud platform. This is an important issue because, the data/service storage/running location specific information is abstracted from the users in Cloud environments.
- Trustworthiness of CSPs: Users are concerned about the trustworthiness of the CSPs. This aspect is different from security because, trustworthiness conveys information pertaining to the task execution such as adhering to Service-Level Agreements (SLA

adherence) and reliability of task execution (such as handling node failure, meeting task deadline etc).

- Dealing with lock-in: In economics, vendor lock-in makes a customer dependent on a vendor for specific products and/or services making it difficult for users to choose another CSP without substantial switching costs. The switching cost includes possible end-of-contract penalties, charges for format conversion and data/application switching and possible additional charges for bandwidth usage.

CSPs Perspective:

- Understanding the market: New Cloud providers may need to understand the current market status in terms of the competitors in the domain, the user preferences in terms of the products/services they prefer most of the time, user preferences for various features such as security and trust requirements etc.
- Adapting to the market: Current Cloud platforms follow a fixed price per resource for their products and services with some small exceptions like Amazon spot pricing. Dynamic pricing strategies are required to improve their performance and to attract more customers based on the market situation.
- Monitoring user profile: With competition among different providers, CSPs may be required to monitor the reliability of users in terms of the feedback given by them to decide user acceptance criteria. It also helps to avoid any unhealthy competition among the providers and users.
- How to attract maximum number of users to use my services. By giving different offers, schemes, discounts, etc.
- How should I adjust my behavior to have a good reputation in the market as well as try to maximize my profits.

II. Background

Literature referred:

Broker-Mediated Multiple-Cloud Orchestration Mechanisms for Cloud Computing, thesis by Dr. Ganesh Iyer. [3]

User's Utility Function

Depends on:

- Trust without Reference (Local Trust)
- Trust with Reference
- Risk-bias of users

Local Trust

It is calculated using the Job Rating given by the user to the CSP for all the past transactions.
More weightage is given to the latest transaction's rating, using a decaying factor α (0 < α < 1).

$$TR_{i,j} = \frac{\sum_{n=1}^{f} \alpha^n JR_{i,j}^n}{\sum_{n=1}^{f} \alpha^n} \in [0, 1]$$

Trust with Reference

It is based also on the trust value assigned by other users, apart from individual trust value.
Since every user's opinion may not be of equal importance, we first calculate Reference Credit for other users, which is a measure of trustworthiness of the reference user.

$$RC_{i,\gamma}^{n} = 1 - |JR_{i,j}^{n} - TR_{\gamma,j}|$$

$$RC_{i,\gamma} = \frac{\sum_{n=1}^{f} \beta^{n} RC_{i,j}^{n}}{\sum_{n=1}^{f} \beta^{n}}$$

Using other user's opinion helps in avoiding bias judgment and to give more trustworthy evaluation of CSP.
Once we get Reference Credit and Local Trust, we now calculate Trust with Reference.
The product of Local Trust of reference user to CSP and the Reference Credit of that reference user is calculated for all the reference users.
This value is used along with the trust value (local trust) given by individual user.

$$TR_{i,j}^{r} = \frac{TR_{i,j} + \sum_{n=1}^{f} RC_{i,\gamma}^{n} TR_{\gamma,j}}{1 + \sum_{n=1}^{f} RC_{i,\gamma}^{n}}$$

Risk-bias of user

Using principles of Von Neumann-Morgenstern utility theorem (VNM-UT), we decide risk-bias of the user.
In general, we can categorize users into risk-averse, risk-seeking or risk-neutral.

Risk averse users are reluctant to accept an offer with an uncertain payoff compared to a certain but lower payoff.
Risk seeking users prefers to take risks.
Risk neutral users are those who do not fall into above two classes.

Utility Function

$$U(TR_{i,j}^{r}) = \frac{1 - e^{-\lambda TR_{i,j}^{r}}}{1 - e^{-\lambda}} \cdot \frac{1}{E_c}$$

The utility function takes into account risk, trust and cost parameters.
User chooses the CSP which maximizes the expected utility value.

Dynamic Pricing by CSPs:

The new price depends on a number of factors:

- Previous price P t-1
- Market Price of a resource maintained by the Broker Q t-1
- Acceptance rate of CSP A-t

Acceptance rate: Ratio of total resources Sold and total resources Offered.

There is a Epsilon which is a weighting parameter to regulate between previous price and market price. The CSP adjusts its price around the price it offered the last time to this user and the average price offered by all the CSPs Q_{t-1} according to the supply and demand of its offered resources. If $A_t > A_{thr}$, it means the CSP's offerings have been accepted adequately and it can choose to increase its price. If $A_t < A_{thr}$, it means the CSP's have been rejected too much and it should consider to reduce its price to win more users. ξ is the parameter indicating how much the CSP want to refer to its own price or the average price offered by all the users in last transaction.

Drawbacks:

- Reputation of CSPs should be considered while computing the price dynamically. Market reputations of CSPs to be maintained globally by the Broker, evaluated based on the Job Ratings CSPs get from users, CSPs throughput and CSPs affinity value.
- Resource Popularity is not considered. In real world, it’s mostly seen as a resource becomes more and more popular, it’s supply increases to meet high demands, and thus the value falls due to high competition.
- Competition in Cloud market should also be considered. Somehow CSPs should also notice the prices other CSPs provide.

III. Proposed Model

Our model uses the same utility function, but a different dynamic pricing method.

Popularity of a Resource[2]
Idea:
- Each CSP maintains statistics of most viewed or requested resource.
- Adjust the price for the resources which are very high currently in the market.

Competition in the market[1]

Idea:

- Every CSP is competing against the rest in the market.
- It maintains the price offered by other CSPs for different resources and it adjusts its pricing in order to stay in the competition.

Fall in Reputation

Idea:

- If the reputation of a CSP changes then he has to vary his prices in order to restore his reputation or gaining maximum profit.
- If his reputation decreases then he has to decrease his prices.
- If his reputation increases then he will increase his prices to gain maximum profit.

Factors:

- Previous price offered by it.
- Market price of the requested resource - Summation of product of CSP's reputation and average price that CSP is offering for a particular resource.

$$AvgPrice = \sum_{k=1}^{n} R_k * P_{kj}$$

- Acceptance rate: Ratio of total resources Sold and total resources Offered.
- Acceptance threshold - This is the threshold reputation value that a CSP wish to maintain.
  If Acceptance rate > Acceptance threshold, CSP increases price
  If Acceptance rate < Acceptance threshold, CSP reduces price.
- Resource Popularity - It's a metric to determine which resource is in demand currently in the market.

$$P_t = P_{t-1} + \frac{((AvgPrice + P_{t-1})/2) * (AcceptanceRatio - AcceptanceTh) * (e^{R_{current} - R_{th}})}{ResourcePopularity}$$

IV. Performance Evaluation

Simulation Details:

- Languages used:
    - C++, to implement our main model and previous model.
    - Python, scripts to do basic tasks such as storing initial data and other data chunks into a persistent dictionary so that they can be used again, as Randomly generated values will be set to new Random values on compiling the project again and again.
- Comparison Parameter used between any two models:
    - Revenue earned by CSPs
    - Jain's Fairness Index
- Model compares with the previously proposed model in our referred thesis by Dr. Ganesh Iyer. It aims to enhance the model.

Simulation Setup:

In our simulation setup, there are 5 users, 5 resources and 5 CSPs in the market. The following table summarizes the initial configuration:

Users	5
CSPs	5
Resources	5
Range of Job Rating	[0.5,0.75]
Range of Budget	[40,100]
Range of Reputation	[0.5, 0.55]
Total Number of Iterations	1000
Frequency of application of	100

Dynamic Pricing (iterations)	
Alpha, value for user’s dependence on previous Job Ratings with time	0.83

Initial configuration of each CSP:

CSP : 1
Reputation Threshold: 0.54
Acceptance Rate Threshold: 0.63
User Resource Price Data

6	5	11	12	6
16	13	17	14	7
10	7	18	12	15
19	17	17	8	19
17	18	6	6	12

CSP: 2
Reputation Threshold: 0.54
Acceptance Rate Threshold : 0.63
User Resource Price Data:

11	19	19	12	13
19	10	5	13	6
6	10	16	8	7
10	6	6	5	5
19	17	16	10	6

CSP: 3
Reputation Threshold: 0.54
Acceptance Rate Threshold: 0.62
User Resource Price Data:

7	14	5	17	5
19	17	19	19	8
10	12	8	11	13
13	14	8	16	19
9	17	5	16	8

CSP: 4
Reputation Threshold: 0.54
Acceptance Rate Threshold: 0.61
User Resource Price Data:

6	11	14	19	5
15	19	18	7	11
7	7	17	17	12
16	6	10	16	6
5	19	9	19	15

CSP: 5
Reputation Threshold: 0.53
Acceptance Rate Threshold: 0.65
User Resource Price Data:

12	17	16	11	10

14	12	17	6	7
8	11	10	6	13
19	19	18	11	7
19	8	19	5	8

RESULTS

1. Effect of Static Job Rating vs Dynamic Job Rating

Job Rating is given by every user to the CSP which it worked with in every iteration. It works as a kind of feedback for the service provided by a CSP. As mentioned previously, Job Rating lies between **(0,1]**.

Case 1: Static Job Rating:

Job Ratings which the users give to the CSPs they got service from, remains the same which is assigned initially. The initial values before any iteration begins is assigned randomly. The users keep giving CSPs the same job rating again and again.

Case 2: Dynamic Job Rating:

As done by our referred thesis, we increase the Job Rating assigned by a user to a CSP after every iteration of market purchases. We increase it by **0.001** after every iteration, until the value reaches 0.98 when it stays the same.

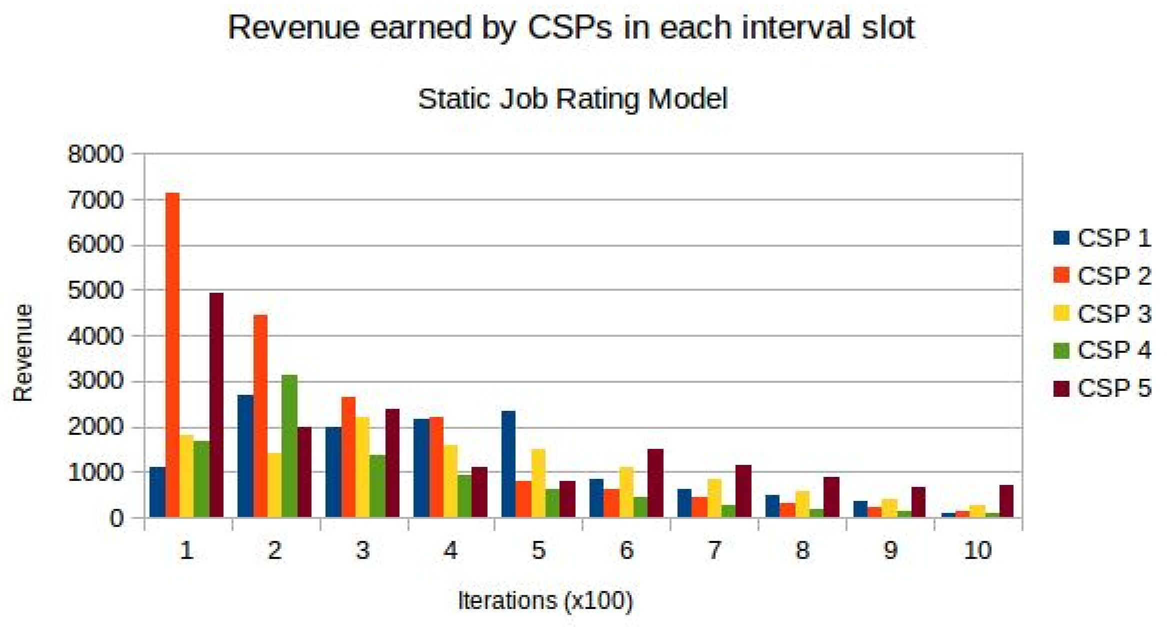


Fig. 1:: Case 1: Static Job Rating

Static Job Rating does not help CSPs to improve, as they keep getting the same rating iteration after iteration.

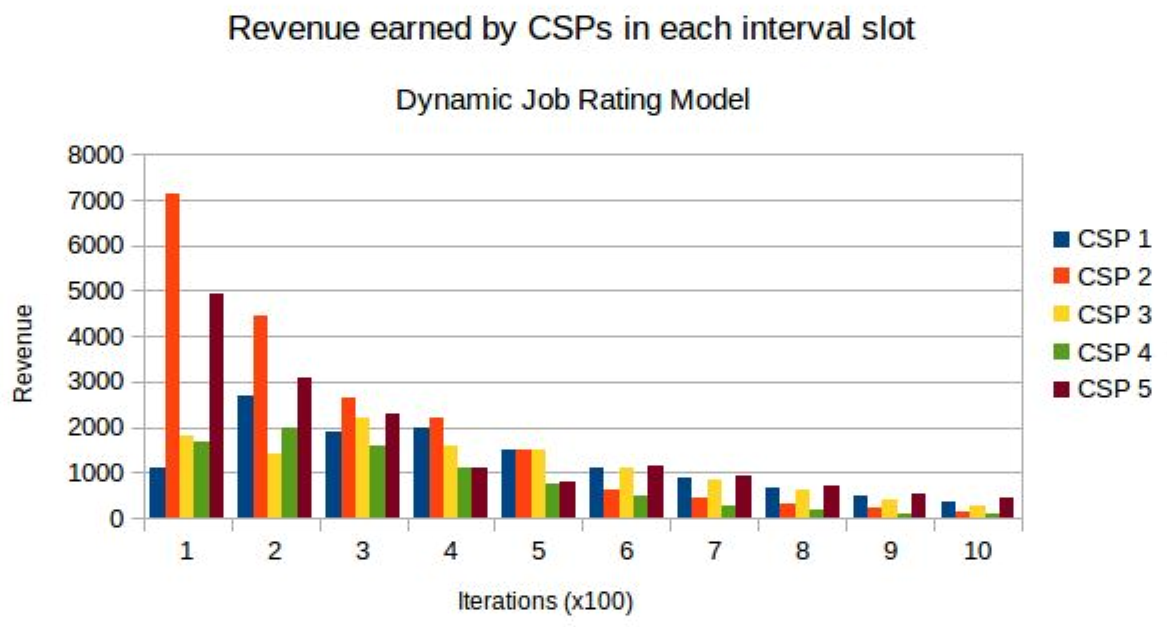


Fig. 2:: Case 2: Dynamic Job Rating

As seen, in case of Dynamic Job Rating CSPs are getting to improve upon increasing Job Ratings. The utility value for a CSP is dependent on the Job Rating which CSP gets, therefore good chances lies that with good feedback CSP can improve more.

The performance, of both are more explicitly compared using Jain's Fairness Index. This value portrays the fairness in sharing of revenue amongst CSPs.
This index lies between [0,1], and our aim here is to reach near 1.

As seen in Fig.3 , Jain's index value shows a lot improvement in case of Dynamic Job Rating model.

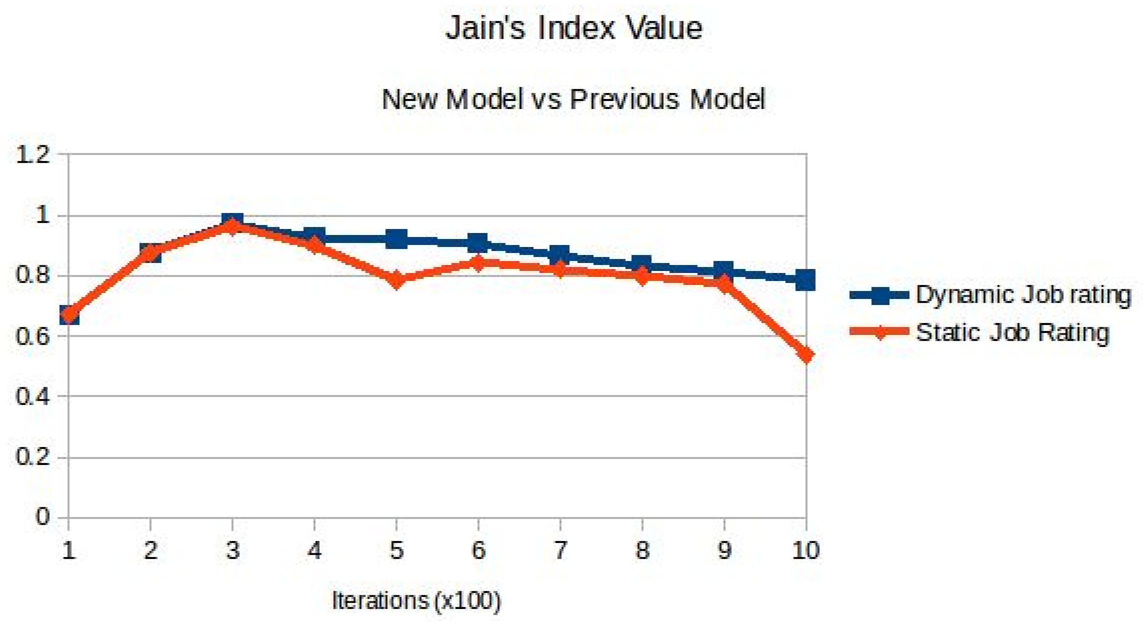


Fig. 3:: Jain's Index Comparison for Case 1 and 2

2. Effect of Static Pricing vs Dynamic Pricing

Case 1: Static Pricing:

After every interval of let’s say **freq=100** iterations, we do not update the pricing offered by CSPs. They remains the same, interval after interval.

Case 2: Dynamic Pricing:

Following the above proposed Dynamic Pricing Strategy function, we update the pricings of every CSP after **freq=100** iterations.

Fig.4 and 5 portrays the results of case 1 and 2 respectively.
In case of static pricing, only the CSPs with minimum price value, which maximises the utility function of users gets chosen every iteration. Other CSPs do not update their prices, and in turn do not get a chance to change the utility perspective of user from them. Thus the CSPs ought to earn good, earns good. CSPs shows not modifications in prices, therefore users ends up choosing same CSPs over and over again.

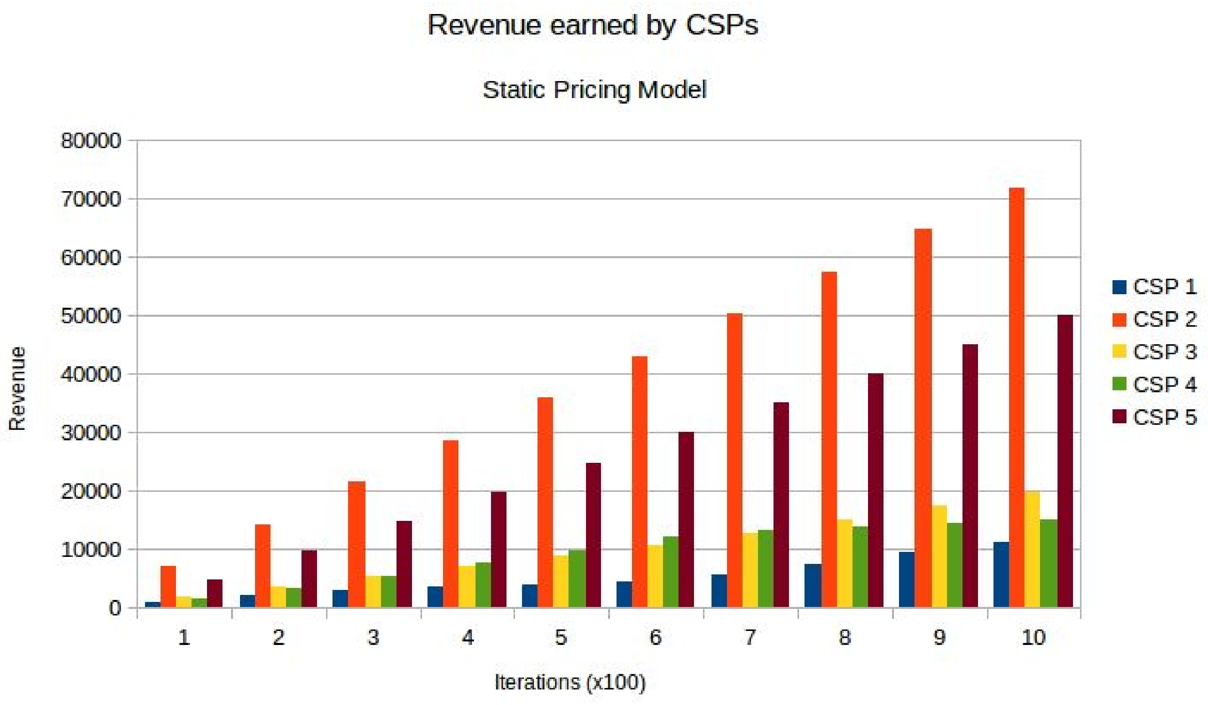


Fig. 4:: Static Pricing Model

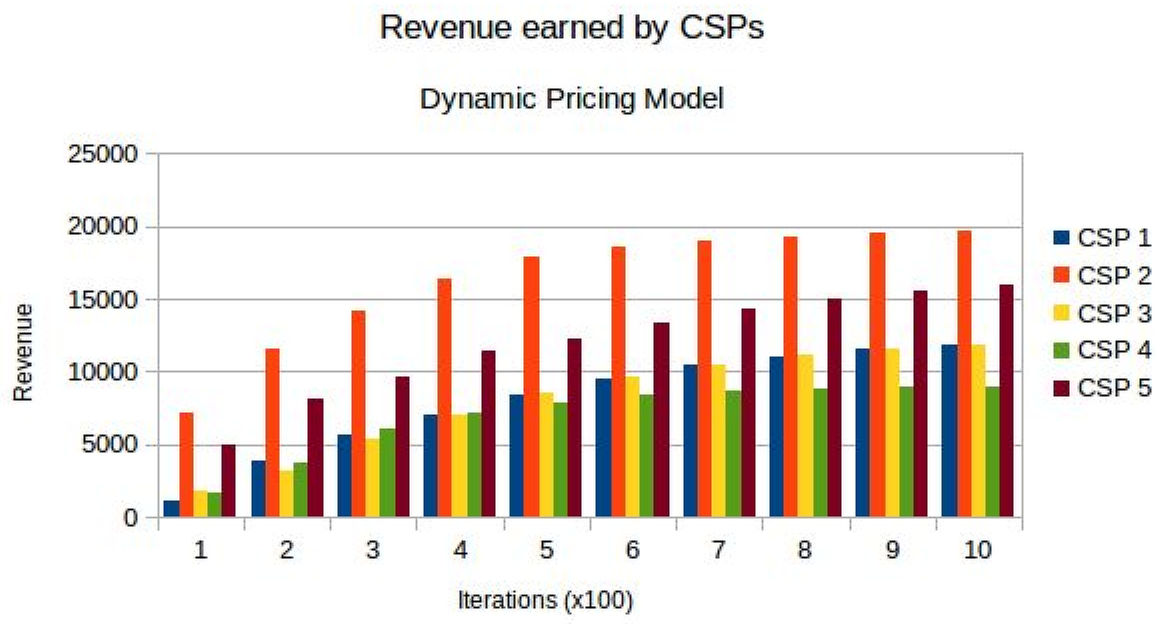


Fig. 5:: Dynamic Pricing Model

In case of Dynamic Pricing, CSPs improve upon their pricing upon various factors mentioned in the Dynamic Pricing Strategy. Thus, update in CSPs prices leads to users switching to other CSPs for service. This leads to improvement in revenue earned by CSPs, can be seen through the results. ***Clearly in Fig.5 CSP 1 improves upon his revenue by updating its prices, whereas in Fig.4 it remains poor throughout till the end.***

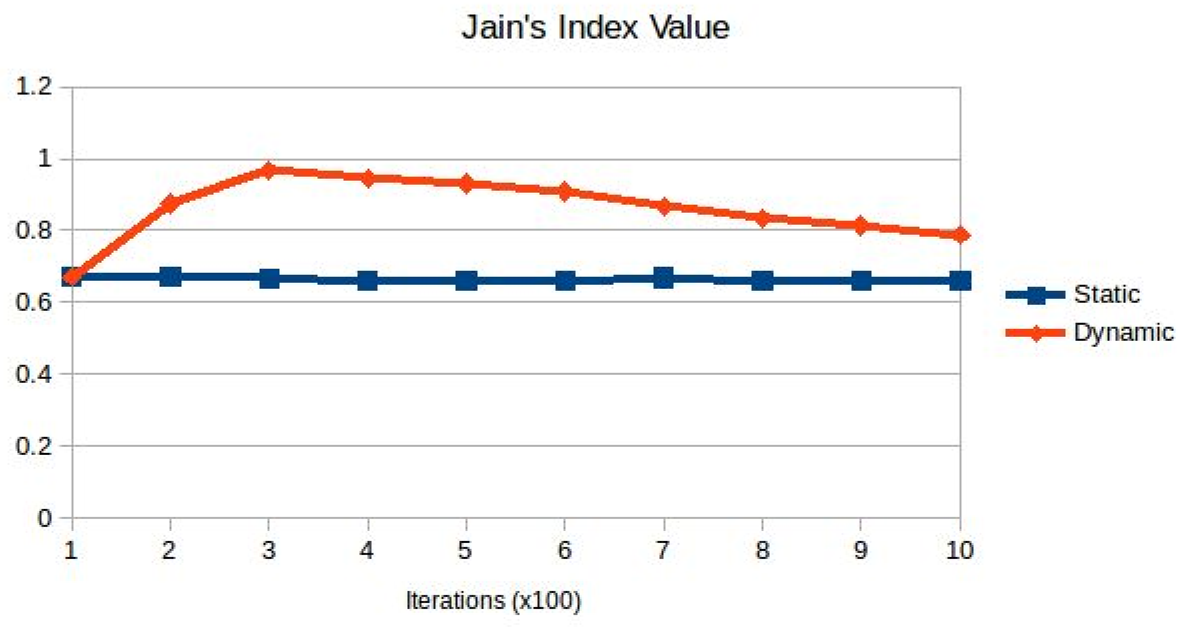


Fig. 6:: Jain’s Index Comparison for Dynamic vs Static Pricing Model

From Fig.6, we can easily conclude that Dynamic Pricing allows CSPs to improve.

3. Effect of ‘rho’ ρ

Here, we analyse different values of ‘rho’ and the results they give. Our ultimate aim here is to bring uniformity to the market so that users get minimal prices and CSPs could earn maximum profit at the same time as well. Thus we would aim to choose such rho which gives us the best Jain’s Fairness Index.

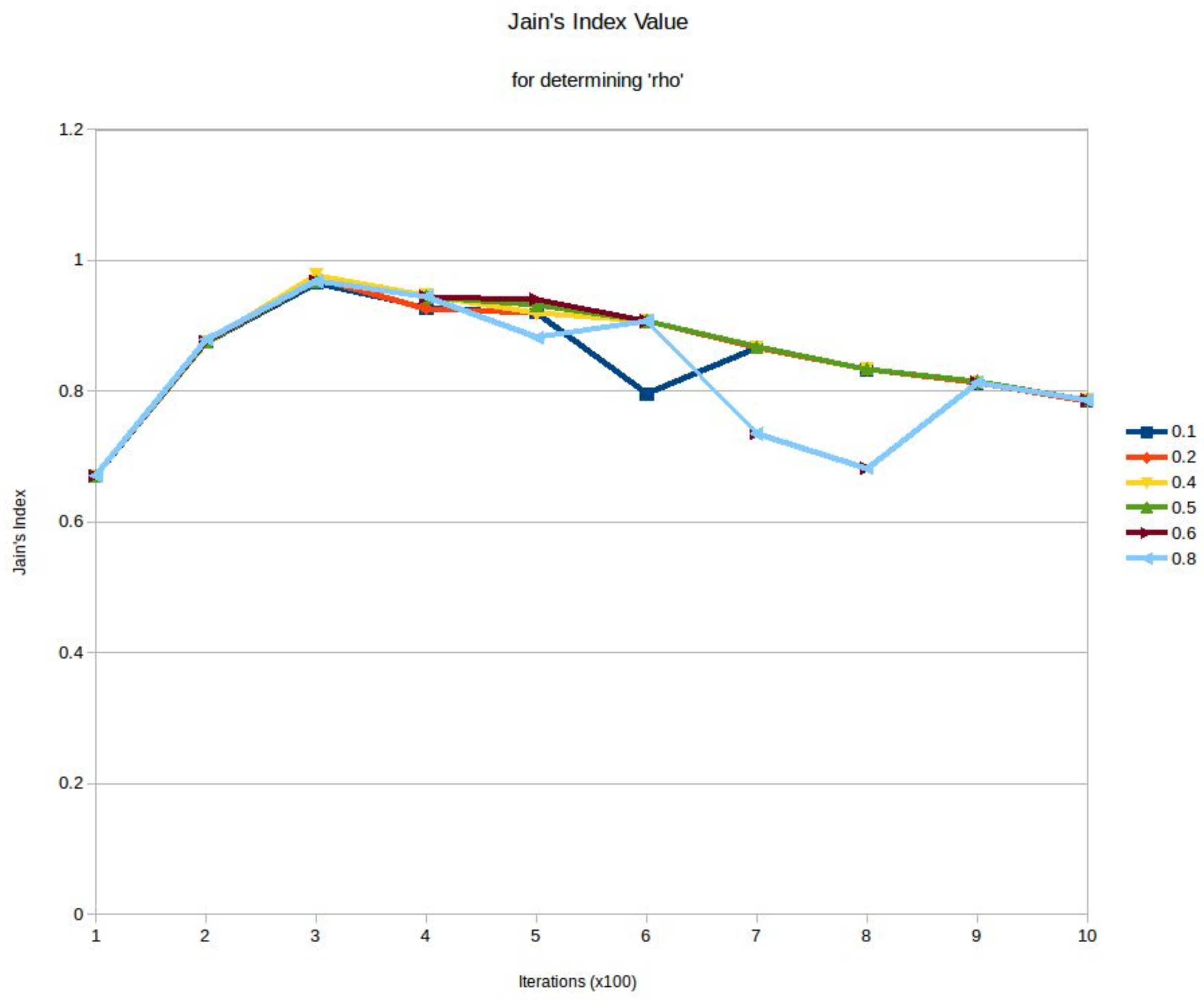


Fig. 7:: Jain’s Index Comparison for different values of ‘rho’

The following table displays the Jain’s fairness index value for different values of ‘rho’:

0.1	**0.2**	**0.4**	**0.5**	**0.6**	**0.8**
0.670563	0.670563	0.670563	**0.670563**	0.670563	0.670563
0.875533	0.875538	0.875548	**0.875553**	0.875565	0.879503
0.965742	0.97341	0.977581	**0.967933**	0.968033	0.968117
0.927065	0.926793	0.9464	**0.944756**	0.944747	0.944834
0.920345	0.919865	0.919243	**0.931203**	0.940981	0.882134

0.795491	0.907111	0.907717	**0.907721**	0.907373	0.906963
0.866852	0.866795	0.867711	**0.867813**	0.735897	0.73499
0.833198	0.833076	0.834225	**0.834423**	0.68235	0.682008
0.812654	0.812544	0.814109	**0.814517**	0.813934	0.813475
0.785353	0.784856	0.786427	**0.787101**	0.786537	0.785891

We choose 'rho' as 0.5 as it gives us the best possible Jain's Fairness Index configuration.

4. Effect of frequency of applying Dynamic Pricing Strategy

Case 1: We apply dynamic pricing after every 100 iterations, for 1000 iterations.

Case 2: We apply dynamic pricing after every 200 iterations, for 1000 iterations.

As, we can see from Fig 8 and 9 below, when dynamic pricing is applied more frequently more is the chance CSP gets to improve upon its price depending on the market parameters and situation. As clearly visible, in first case CSP5 easily overtakes CSP2 in terms of revenue gained, as it gets more chance to update its dynamic value as compared in second case, where it might require more iterations for that to happen.

Even Fig.10: Comparison based on Jain's Index also shows that Case-1 works better.

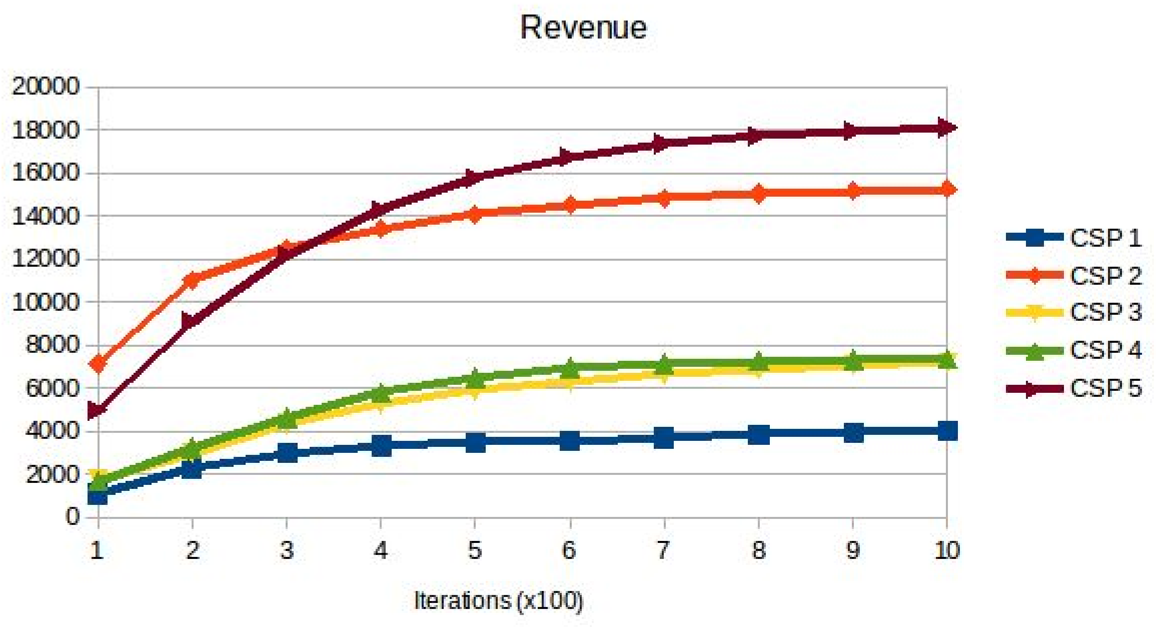


Fig. 8:: Frequency: 100 iterations

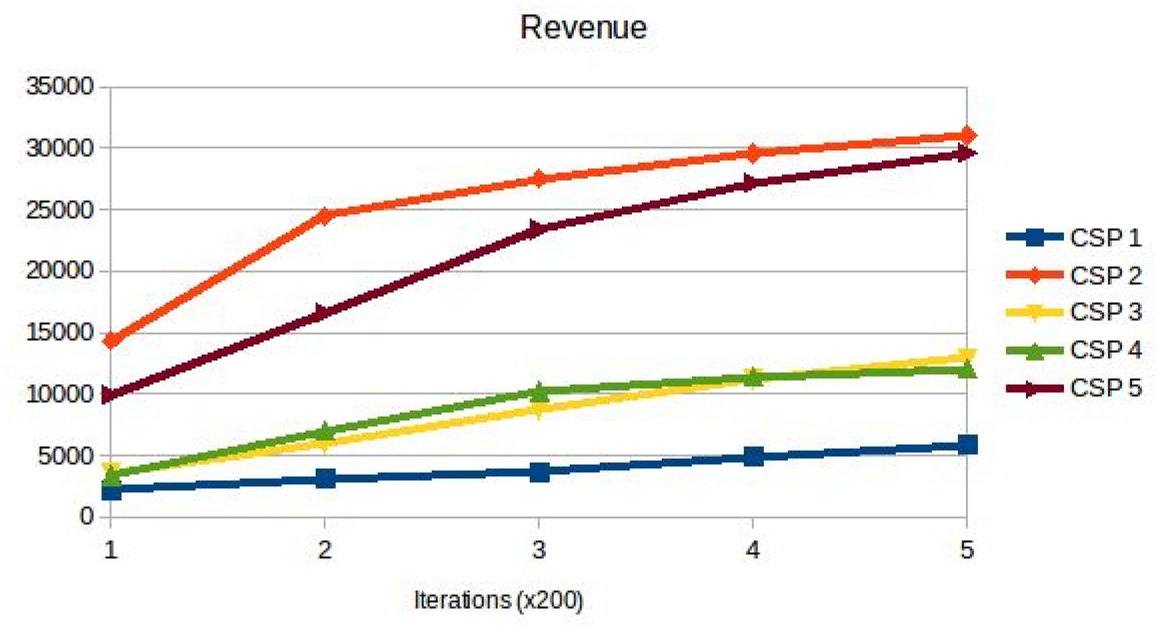


Fig. 9:: Frequency: 200 iterations

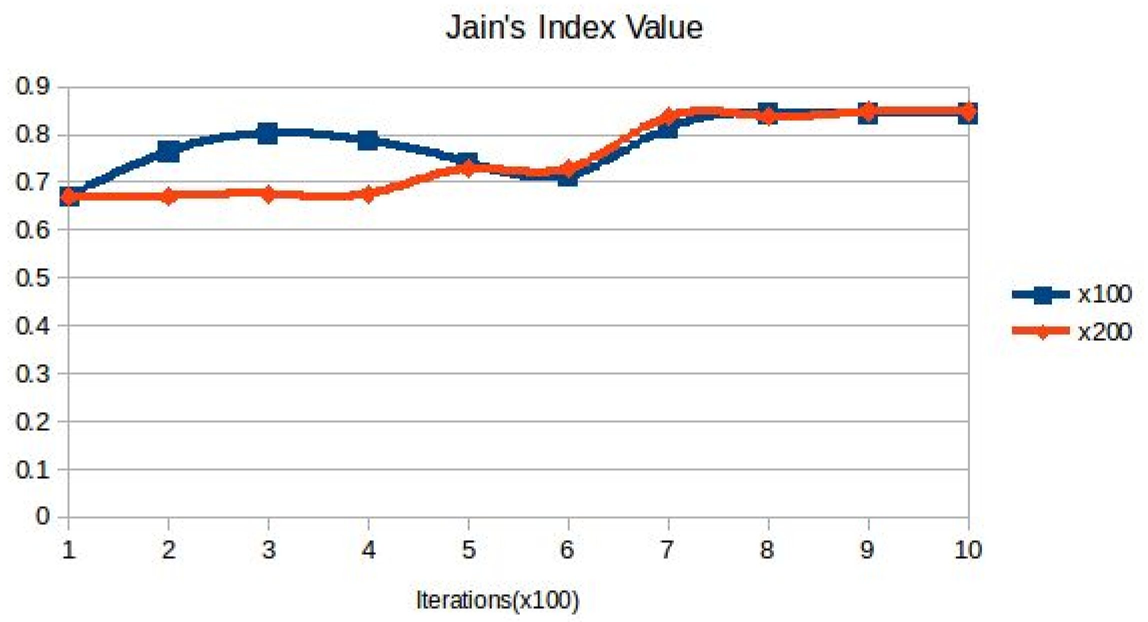


Fig. 10:: Jain's Index Comparison for Case 1 vs Case 2

In real case, it depends on the CSP's willingness. If the CSP is satisfied with its performance, it can slow its adjustment with the market and vice versa. An important conclusion from this study is that CSPs need not monitor and alter the price offer after every iteration. Even if they alter the price offers periodically, they can still adapt to the market conditions quickly.

5. Comparison with previous model

There are many factors which govern the differences between the two model's results. So, they can't be directly reasoned out.

But in general, the new models aims to bring uniformity in the market. It considers supply-demand factor, promotes competition in the market so that the prices fall uniformly to improve upon user's utility and his/her budget, simultaneously aiming to increase CSPs profits.

Case 1: Previous Model

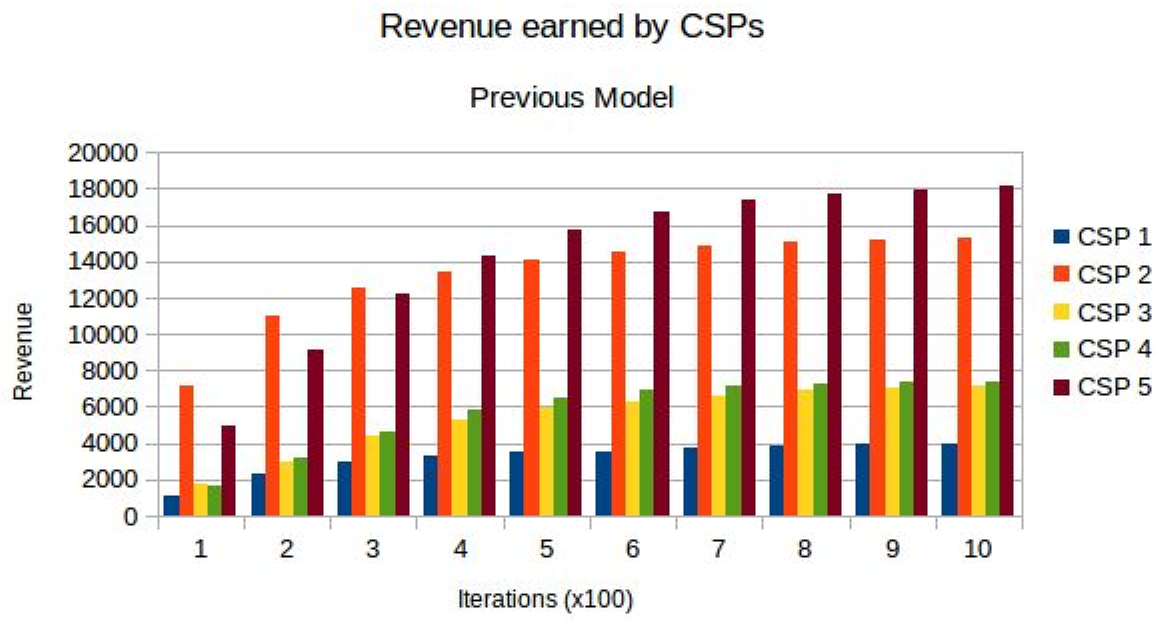


Fig. 11:: Previous Model's application

Based on the CSPs Revenue vs Iterations (x100) data:

CSP 1	CSP 2	CSP 3	CSP 4	CSP 5
1089	7128	1782	1683	4950
2289	10997.4	2982	3216.33	9083.33
2971.67	12496.8	4359.78	4622.56	12194.4
3327.22	13392.4	5278.3	5784.04	14268.5
3477.15	14070.2	5890.64	6475.4	15738.3
3542.99	14522	6298.87	6936.3	16752.3
3692.33	14823.2	6640.21	7149.28	17348
3838.64	15024	6897.73	7248.78	17710.9

3936.19	15157.9	7069.41	7315.11	17952.8
4001.22	15247.1	7183.86	7359.33	18114.1

Case 2: New Model

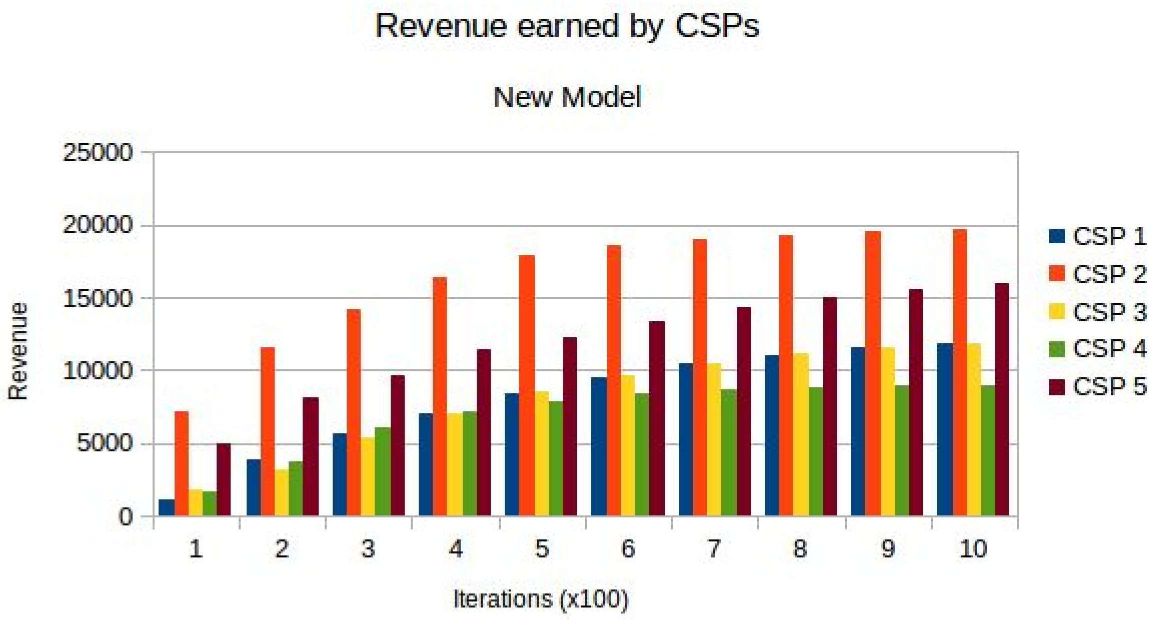


Fig. 12:: New Model's application

Based on the CSPs Revenue vs Iterations (x100) data:

CSP 1	CSP 2	CSP 3	CSP 4	CSP 5
1089	7128	1782	1683	4950
3798.5	11557.4	3199.19	3674.2	8055.29
5688.65	14186	5397.19	6050.93	9573.13
6999.23	16415.3	7002.16	7159.02	11357.8
8406.07	17910.6	8527.05	7901.99	12244.4

9523.72	18522.2	9654.52	8397.3	13405.4
10389.2	18960	10503	8670.1	14313.3
11043.9	19264.9	11117.3	8825.85	15020.4
11522.6	19473.6	11538.5	8929.69	15567.4
11866.5	19612.7	11819.2	8998.92	15993.7

The two models can be compared on Jain's Fairness Index, and the results are as follows:

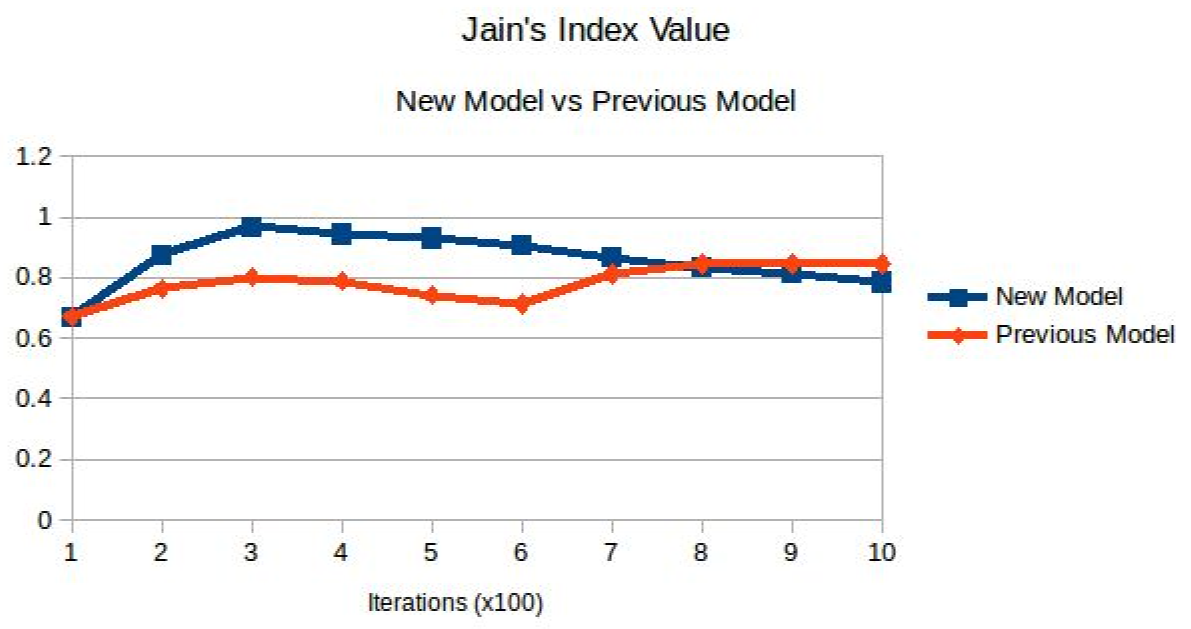


Fig. 13:: Jain's Index Comparison for the two models

From the data:

New Model	Previous Model
0.670563	0.670563
0.875553	0.764163

0.967933	0.802026
0.944756	0.787195
0.931203	0.742091
0.907721	0.713454
0.867813	0.813599
0.834423	0.844522
0.814517	0.844522
0.787101	0.844522

Thus, using Jain's Fairness Index as a comparison parameter we find New Model to be better in initial cases of market transactions pushing CSPs towards a competitive pricing model to distribute revenues fairly.

V. Conclusion

Thus, we conclude from the results obtained from simulations that the proposed new model works as an enhancement to the previous model, and gives improved results. It considers many other factors which affects the market and thus is able to perform better at many places.
Considering Market Competition, Market Reputation and Resource Popularity in addition helps it to improve the Fairness Index Value.

VI. References

[1]
http://alumni.media.mit.edu/~lysi/Papers/IJISAFM_dynamic_pricing.pdf

[2] http://www.cio.com/article/2870961/consumer-technology/report-analyzes-amazons-dynamic-pricing-strategy.html

[3]
Thesis on “Broker-Mediated Multiple-Cloud Orchestration Mechanisms for Cloud Computing” by Dr. Ganesh Iyer

---

VII. Appendix

Code Repository:
https://github.com/Ammy2/BTech-Project-Risk-based-Cloud-Broker-System

---