

AI based Smart Meter for Future Homes

Sarthak Jain, Aditya Mantri, Nachiket Trivedi, Nishant Jani
Department of Computer Engineering, San Jose State University
San Jose, California

Abstract—As we know that power consumption is increasing day by day, so we need to create technologies that can help us in utilizing our resources wisely. Here we have proposed a technique using the particle swarm algorithm through which one can monitor the power consumption of each of their devices based on which it provides the optimal schedule to utilize that device keeping in mind all the devices installed at a particular home. The overall objective is to schedule multiple devices of the entire house in such a way that the total power consumption of the house in the peak hours is minimized. The user would get charged a minimal amount for their electricity consumption.

Index Terms—particle swarm algorithm

I. INTRODUCTION

In today's world, the consumption of electricity has been increasing very quickly, and the resources which are utilized for its production are depleting rapidly. Thus due to this gap, the charges per unit of electricity are increasing every year. Many significant power-producing companies find it challenging to cope up with this massive demand for electricity. We have suggested an approach which helps in reducing the overall electricity consumption of a particular home during peak hours by scheduling some devices to run during non-peak hours. The reason behind this is that during peak hours, the charge of electricity per unit is maximum while that during the non-peak hours is significantly low. Thus if a user wants to utilize an appliance, but it's not that urgent to use, then he/she can schedule it to run later on so that the cost of running the same device would reduce significantly than when it would run during peak hours. For example:- the Washing machine is a ubiquitous appliance that everyone uses in their day to day use. It is an appliance that utilizes quite a lot of power to work, so running that appliance during the peak hours when the rate is maximum would charge the user highly as compared to utilizing the same machine during non-peak hours. Our algorithm provides the user with this result where they can get to schedule multiple appliances in such a way that the overall cost occurring to the user gets reduced.

II. METHODOLOGY FOLLOWED

We will be providing the user with smart, IoT based devices like smart plugs and a tablet. The smart plugs would be controlling whether a particular device should be enabled or disabled as per their schedule computed based on the algorithm. The tablet would act as an interface to the user wherein the user would be able to add the list of devices in his home, and then that data would be computed on our algorithm in the cloud. Then the results would be displayed to the user

on their device, and once the user enables scheduling mode for that device, the system would automatically schedule the device based on the time which the algorithm had provided it.

A. Technology and tools used

- IBM Cloud
- Particle Swarm Algorithm
- Apache Kafka
- Docker (containerization of application)
- Python Flask
- ReactJS and NodeJS (web application)
- MongoDB (Database)

III. WHAT IS PARTICLE SWARM ALGORITHM?

Particle swarm is an optimization algorithm that performs iterative computations on a given problem equation, which it internally creates, and then based on that equation, it provides the final result as to which parameter selection would lead to the most optimal answer. It approaches a problem by considering various solutions of a given problem that are here considered to be particles and then moving these particles around the search space in order to find out a solution to a problem based on a mathematical equation with particle's position and velocity as parameters. The search space can vary, but still there is no effect on the final output of the result obtained.

Each particle has its own search space, which would provide a locally optimal solution while in the bigger picture, we have multiple particles, so we should consider global search space in order to reach a global optimum answer. Here we have the problem of finding the optimal schedule for each of the devices which are used by a particular user. Thus we consider various devices as particles, and then we find a global optimal answer which would provide us the minimum cost for operation of each device based on the schedule the algorithm has computed. Thus we frame the mathematical equation here in such a way that the total cost of operation of the devices turns out to be minimum by varying various parameters like different schedules, which in turn would be having a different cost per 1 unit of electricity. So the algorithm would perform various combinations of all the schedules and then come to a solution that would deliver the minimum electricity bill.

A. Algorithm

As we had previously stated that various potential solutions to our problem act as a particle and then based on the

mathematical equation of the particle's position and velocity optimization is performed through an iterative process. The movements of these particles are around their local best search space along with that of the global search space so when improved positions are found then the movement of the particles are along that way.

```

for each particle  $i = 1, \dots, S$  do
  Initialize the particle's position with a uniformly distributed random vector:  $\mathbf{x}_i \sim U(\mathbf{b}_{lo}, \mathbf{b}_{up})$ 
  Initialize the particle's best known position to its initial position:  $\mathbf{p}_i \leftarrow \mathbf{x}_i$ 
  if  $f(\mathbf{p}_i) < f(\mathbf{g})$  then
    update the swarm's best known position:  $\mathbf{g} \leftarrow \mathbf{p}_i$ 
  Initialize the particle's velocity:  $\mathbf{v}_i \sim U(-|\mathbf{b}_{up}-\mathbf{b}_{lo}|, |\mathbf{b}_{up}-\mathbf{b}_{lo}|)$ 
  while a termination criterion is not met do:
    for each particle  $i = 1, \dots, S$  do
      for each dimension  $d = 1, \dots, n$  do
        Pick random numbers:  $r_p, r_g \sim U(0,1)$ 
        Update the particle's velocity:  $\mathbf{v}_{i,d} \leftarrow \omega \mathbf{v}_{i,d} + \varphi_p r_p (\mathbf{p}_{i,d} - \mathbf{x}_{i,d}) + \varphi_g r_g (\mathbf{g}_d - \mathbf{x}_{i,d})$ 
        Update the particle's position:  $\mathbf{x}_i \leftarrow \mathbf{x}_i + \mathbf{v}_i$ 
        if  $f(\mathbf{x}_i) < f(\mathbf{p}_i)$  then
          Update the particle's best known position:  $\mathbf{p}_i \leftarrow \mathbf{x}_i$ 
        if  $f(\mathbf{p}_i) < f(\mathbf{g})$  then
          Update the swarm's best known position:  $\mathbf{g} \leftarrow \mathbf{p}_i$ 

```

Fig. 1. Brief Overview of the Algorithm [1]

The above figure gives a brief overview of how the particle swarm algorithm works and how it comes to an optimal solution. So each particle would be having two search spaces, a local one and another global one. Based on how each of these search spaces better performs, the particle moves towards that search space. So as each particle aligns themselves towards their best position where they can lead to an optimal solution, all these particles start converging towards a particular area. Then, later on, they all merge into a single point. Here instead of having a position of a point as a parameter, we have timing/schedule for running a device as a parameter for the mathematical equation of the problem, which needs to be optimized to get the final solution.

B. Convergence

The algorithm would converge to a single point when we are close to obtaining an optimal solution to our problem.

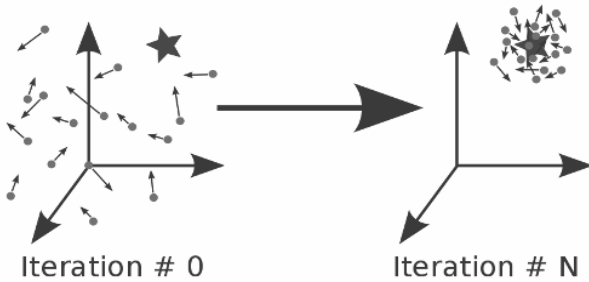


Fig. 2. Convergence of particles [2]

As we can see in the figure stated below that how scattered the particles are initially and how they gradually converge as we perform iterations over the mathematical equation of the problem. Initially, all the particles are separate and in different directions having different velocity and position. However, as each particle starts searching their search space, they get to know where they can find the right solution, so then iteratively,

each particle goes towards an optimal position of the problem. Thus one needs to perform many iterations in order to reach that point.

$$\mathbf{v}_{i+1} = \omega (\mathbf{v}_i + \eta_1 \mathbf{r}_1 \cdot (\mathbf{x}_i - \mathbf{x}_i^l) + \eta_2 \mathbf{r}_2 \cdot (\mathbf{x}_i - \mathbf{x}^g))$$

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{v}_i$$

$$\mathbf{v}_{i+1} = \omega \mathbf{v}_i + \eta_1 \mathbf{r}_1 \cdot (\mathbf{x}_i - \mathbf{x}_i^l) + \eta_2 \mathbf{r}_2 \cdot (\mathbf{x}_i - \mathbf{x}^g)$$

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{v}_i$$

Fig. 3. Standard Mathematical equation used [3]

The above equations represent the standard mathematical equations that are used during the particle swarm algorithm. Here each \mathbf{x}_i represents the i th particle while the \mathbf{v} represents the velocity of that particle. Here we consider time/schedule as a parameter in order to find an optimal solution, so for each device, we check in different time slots what is its power consumption amount generated.

IV. APACHE KAFKA

- It is an open-source stream processing software and is mainly written in Java and Scala.
- It has an immutable commit log from which you can subscribe to it and publish data to any number of systems or applications. It is a highly scalable, fault-tolerant system that is widely used for applications that have a large number of users.

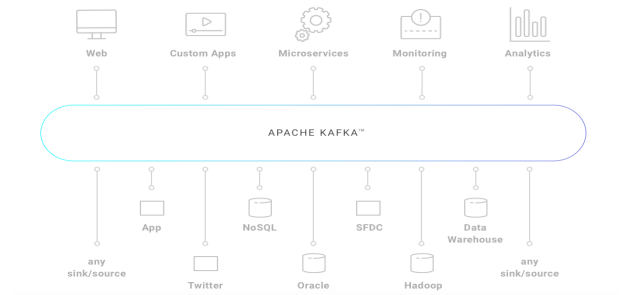


Fig. 4. Overview of uses of Kafka [3]

The above image shows the application of Apache Kafka in various areas. Apache Kafka consists of mainly 4 API's as follows:

- Producer: Gives access to an application to publish streams of records.
- Consumer: Gives access to subscribe to topics and process the streams of records.
- Stream: Converts input stream to output to produce results.
- Connector: Helps in establishing a connection between the consumer of the data stream and then one who

produces it. In our project, we have made the API distributed and robust by integrating Apache Kafka into our backend server.

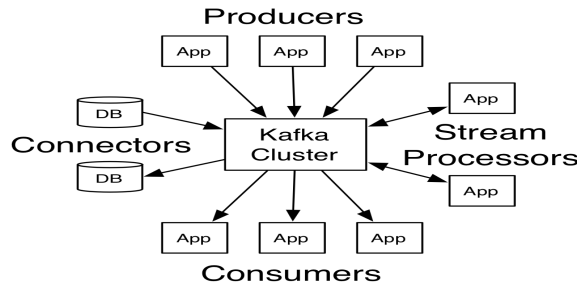


Fig. 5. View of how Kafka working [4]

V. IBM CLOUD

IBM Cloud is the cloud computing services offered by IBM. It includes Infrastructure as a service(IaaS), Platform as a service(PaaS), and Software as a service(SaaS) accessible through different cloud models like public, private, or hybrid cloud.

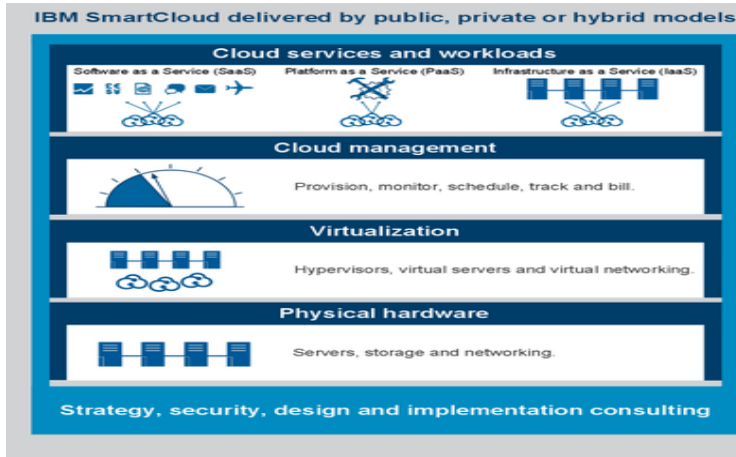


Fig. 6. IBM Cloud Architecture

In the figure above, we can see the various levels of the IBM Cloud. At the topmost level, we have Cloud services and workloads like SaaS, IaaS, and PaaS. Then we have cloud management, which undertakes the task of resource provisioning, monitoring, and billing. Then there is a virtualization layer where there are various hypervisors, virtual servers, and virtual networks. Then at the bottom-most level, we have the physical hardware, which includes the servers, storage, and various networking layers.

A. Application

We have created a web-based application that is hosted on the cloud so that it can easily be accessed from anywhere and even can handle a large number of client requests. We have used NodeJS and ReactJS for creating the web

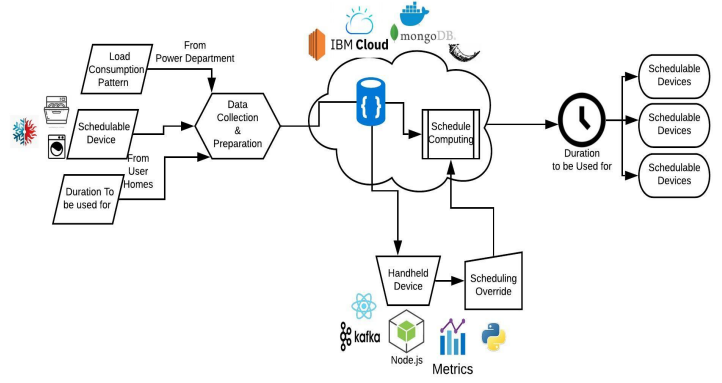


Fig. 7. Architecture Overview

application. The application server wherein the computation using the particle swarm algorithm takes place is hosted on the IBM Cloud. The web application is also hosted on the web application.

The web application has its database like MongoDB, and the application is made to support distributed systems as Apache Kafka has been integrated with it is internal architecture.

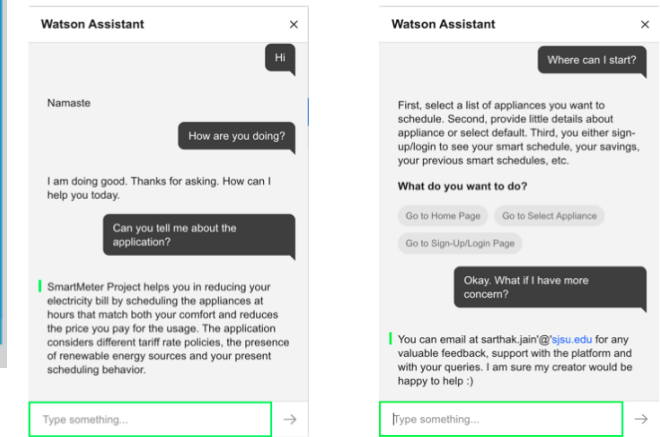


Fig. 8. IBM Watson AI assistant

Fig. 8. shows the AI-based chatbot assistant, which we have integrated into our project. We have made use of IBM Watson for this purpose. It has some predefined answers to some questions so that users can get answers to their questions at any time of the day. The smart chatbot is quite responsive and efficient in providing answers to almost all questions of the users.

The entire application is deployed on IBM Cloud using it's Kubernetes Service. A Kubernetes Cluster is created as can be seen in Fig. 9. This cluster is used to deploy the containerized application both the computational PSO algorithm and the web application.

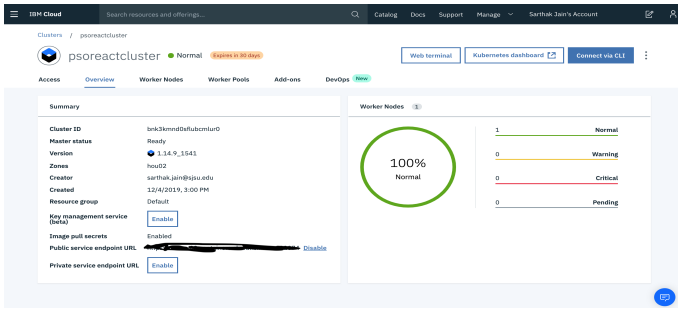


Fig. 9. Kubernetes Cluster created on IBM Cloud

The dockerized application is deployed by first creating a deployment and then a service for both the PSO computational algorithm and the web application. A random NodePort is assigned after successful service creation which can be seen using the kubectl service describe command. Fig. 10 and Fig. 11 shows the docker images present in sarthak_jain namespace in IBM Cloud and the respective Kubernetes dashboard.

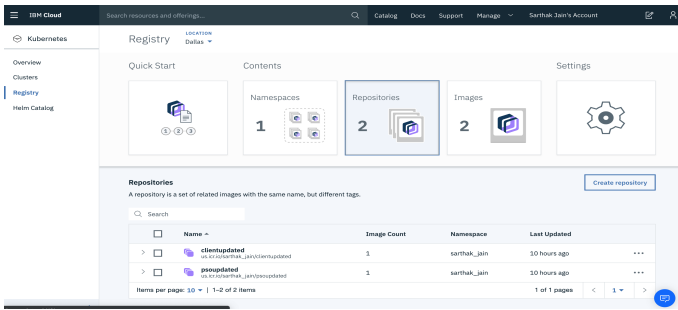


Fig. 10. Docker Images on sarthak_jain namespace in IBM Cloud

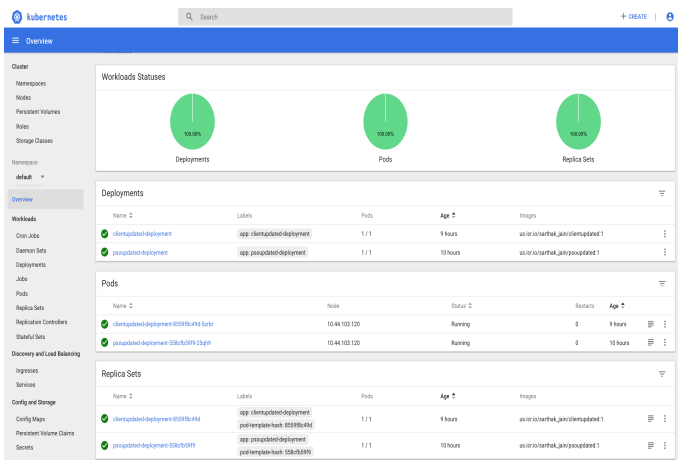


Fig. 11. Kubernetes Dashboard

The user, after landing on to the home page of the application, would see all the services provided by us. The user would then need to provide the device which they have installed in their house. By selecting that device, the user would get redirected to their dashboard. If a user has already been logged

into their account, they would directly get redirected to their dashboard page else; they would be redirected to login into their account first. If the user is a first time user, then they need to register themselves first and then log in to their account. Once the user gets into their dashboard page, they would be able to see all their devices, which they have added. Once they click on the compute now button on the page, their request gets redirected to the cloud server where our particle swarm algorithm is running. The algorithm would perform computation from that device details sent to it by the application, and then after on, it returns the results to the application. The result would show the optimal time at which a device must be run in order to get minimum operational cost for that particular device.

Smart Meter

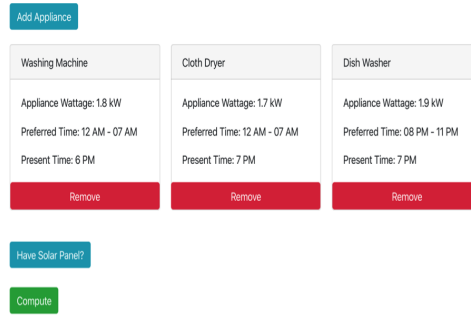


Fig. 12. Adding an appliance

The above photo shows the way the user would be able to see their devices. Here the term device refers to the appliances which the user wants to schedule using our optimization algorithm.

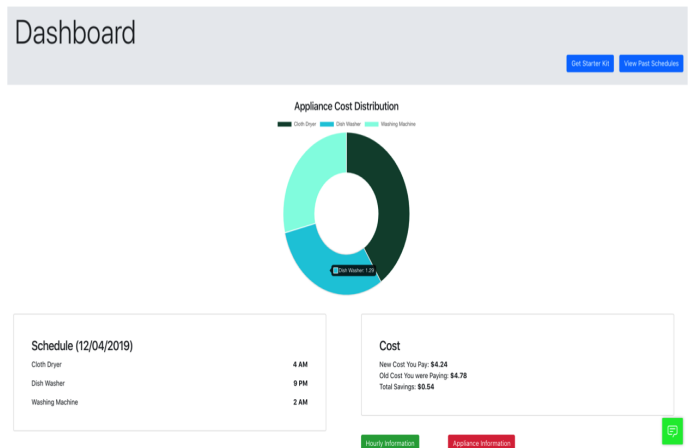


Fig. 13. User dashboard

The user dashboard is shown in the above image. The user would be able to see their devices, which he had decided to

schedule. The dashboard provides the scheduling time of each of their devices along with the total cost incurred by running those devices. It also shows past schedules along with hourly cost information. It shows the current total cost of running those devices also the previous cost of running those same devices. Thus the user can know how much savings does he get. The user can also request a starter kit for their use. The user is provided with appliance information, which shows how much the particular appliance would consume power and that particular appliance's cost for running it.

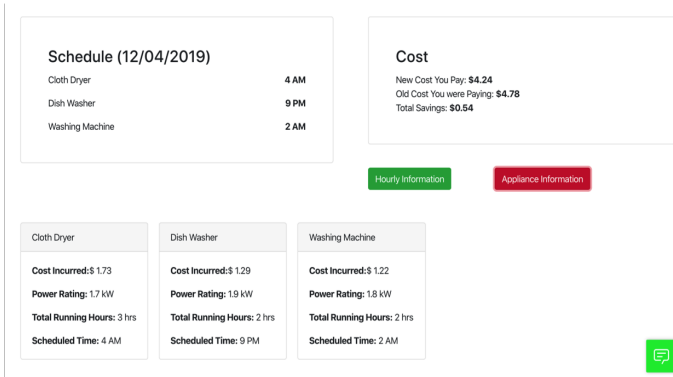


Fig. 14. User dashboard appliance info

The above image shows detailed information regarding each device. Here we can see three devices which are namely: cloth dryer, dishwasher and washing machine. The detailed information provides the cost incurred for running that device along with the device's power consumption, running hours, and the scheduled time for that particular device. This detailed view can provide the user with more detailed information regarding their devices. This can help the user check on which time which particular device is being used so that if a user has priority for a particular device, then they can plan accordingly.

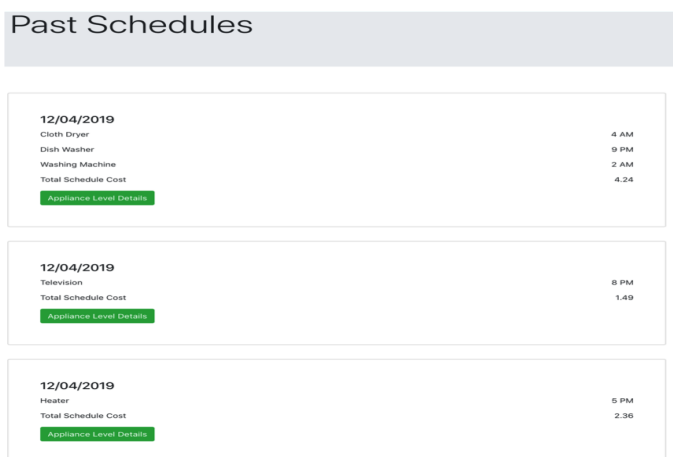


Fig. 15. User past schedules information

The above image shows all the past schedules of the user. So that if a user is planning to schedule their devices, which they had previously scheduled together, they can get an estimate

of the amount of savings they would receive so that they can plan accordingly.

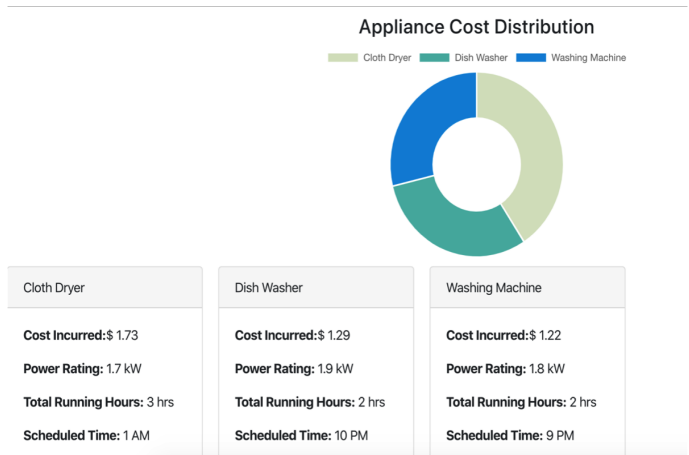


Fig. 16. Detailed past schedules information

The above image shows the detailed cost estimation of the appliances in the past. It shows how much time did those devices were on and what was their previous schedule along with the cost incurred at that time. So that when a user is scheduling a particular device, they can know that what would be its power consumption along with it what would be the amount of cost savings that would take place when they would be scheduling it. The detailed past schedule information can be quite helpful to the end-user and give statistical information to them.

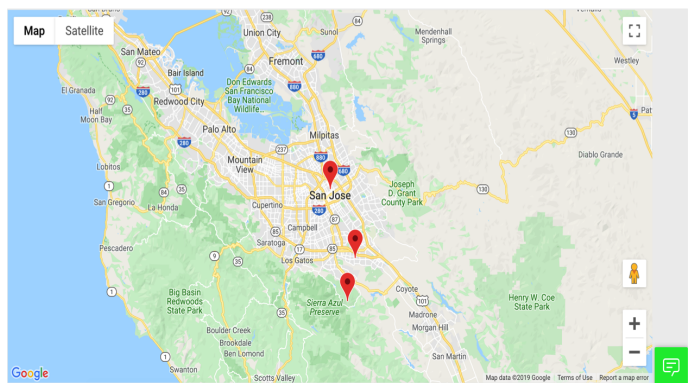


Fig. 17. User Location Information

Fig. 17 shows the present location of all the users of our application. It gives the admin an overview regarding the places where their end users are located.

ACKNOWLEDGMENT

REFERENCES

- [1] https://en.wikipedia.org/wiki/Particle_swarm_optimization
- [2] <https://esa.github.io/pagmo2/docs/cpp/algorithms/psa.html>
- [3] <https://www.confluent.io/what-is-apache-kafka/>
- [4] <https://kafka.apache.org/intro>
- [5] <https://www.google.com/url?sa=source=images&ved=2ahUKEwirx-Sbn6TmAhWxPH0KHxNECDEQjRx6BAGBEA&url=https>