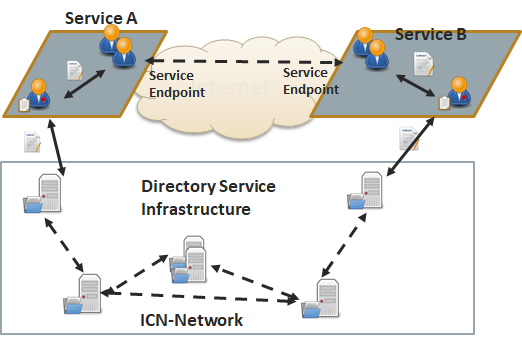
**Cache Allocation Algorithm for Service Directory**

Within the scope of my internship, I worked for a cache allocation algorithm for service directories that help the IoT devices to communicate each other.

First of all, I had to do literature research. The problem we are working on is very similar to the “Uncapacitated Facility Location” problem. So firstly, I tried to understand this problem. In this problem, there are facilities to be opened, and clients that need to be connected to at least one facility. Clients are within a certain distance to each facility. Connecting a client to a far facility reduces the efficiency. On the other hand, there is a cost of opening a facility. Therefore, opening all of the candidate facilities can also reduce efficiency. So, efficiency depends on the number of opened facilities and the distance of customers to their connected facilities. What to do is to minimize the total cost considering the distance of customers to their connected facilities and to determine the most appropriate facility set.

This problem is an NP-Hard problem, and using brute force makes the running time exponential. This method is not feasible when the network is large. Therefore, an optimization algorithm should be proposed.

Figure 1. Uncapacitated Facility Location problem

The problem definition, and adaption to our problem can be found here:

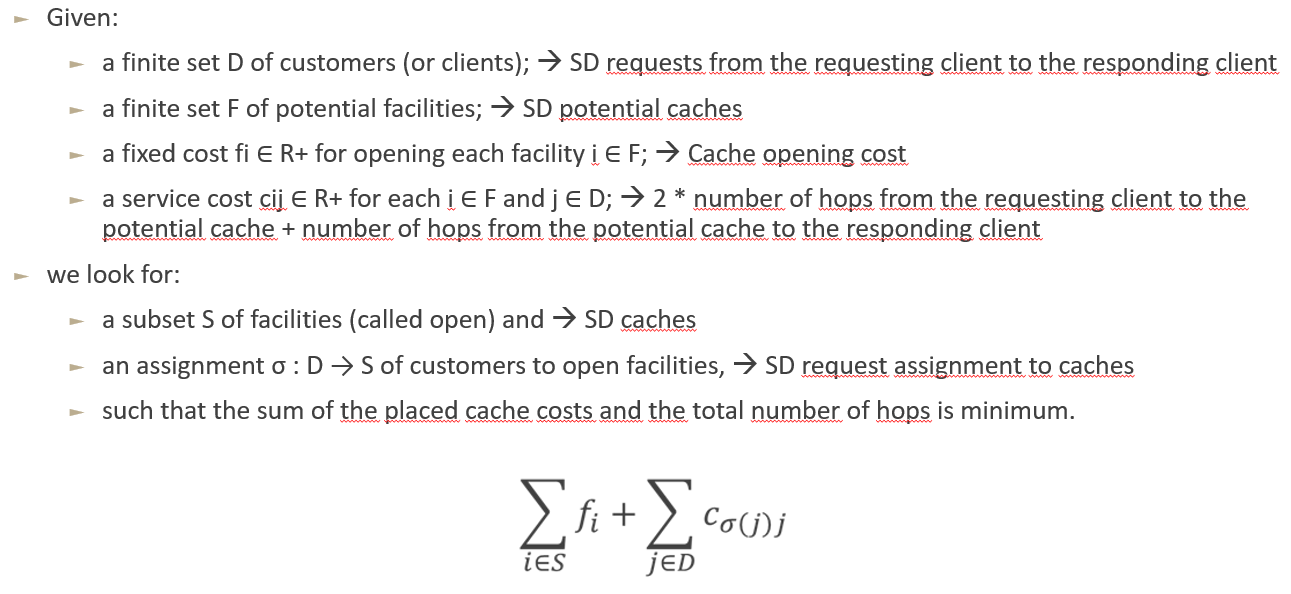


Figure 2. Problem definition

The reason we multiply the first part of the service cost by two is because we want the caches to be closer to the requesting clients rather than the responding clients. The base algorithm can be found in Figure 3.

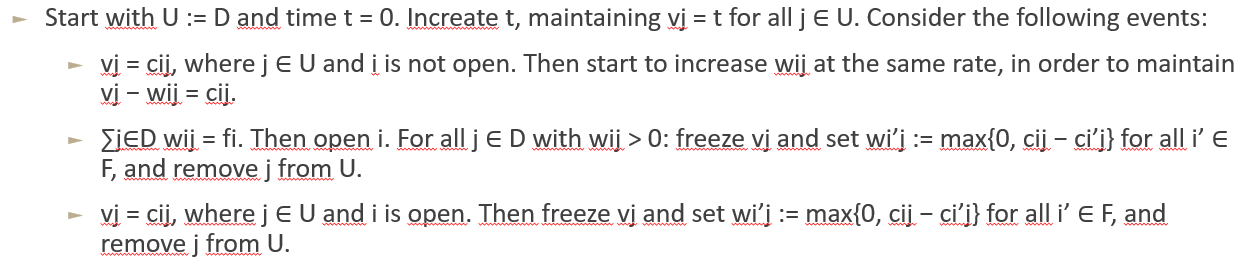


Figure 3. The Jain-Mahdian-Saberi algoritması

For the simulation, I used Ns3. Ns3 is a network simulation tool. A sample screenshot from NetAnim can be found in Figure 5. Also, an offline trace file of this can be found in GitHub repository.

For the experimental results, I did 10 experiments comparing total serving times between our optimized algorithm and random cache placement. As it can be seen from the Figure 6, our optimized algorithm always performs better than the random placement, and it is in fact more stable.

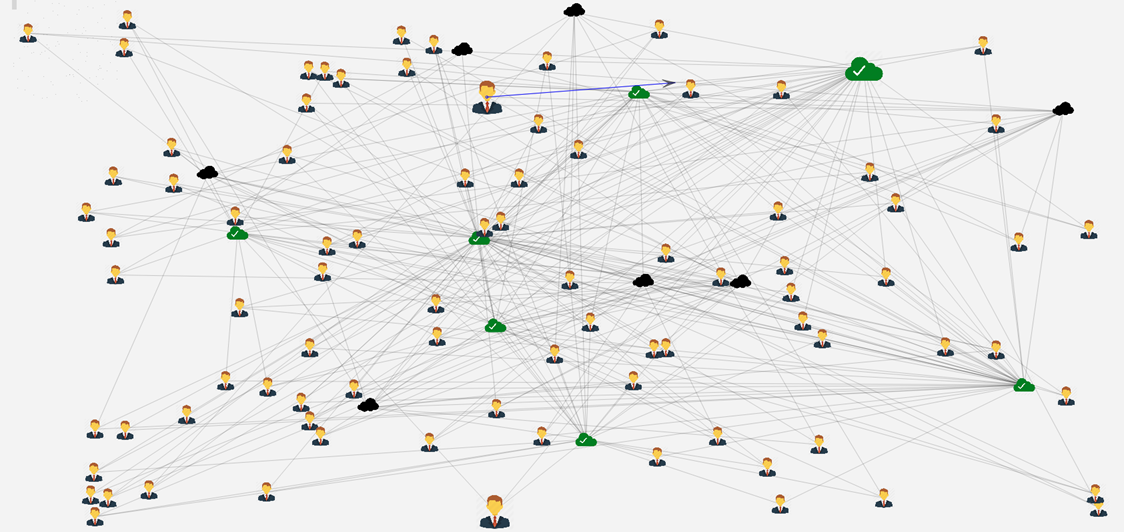
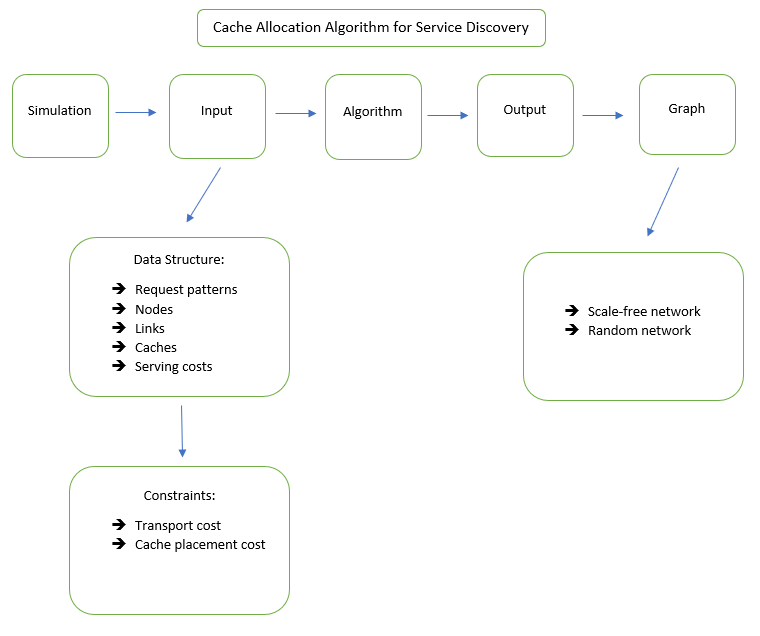
Figure 4. System architecture

Figure 5. A sample simulation scene from NetAnim

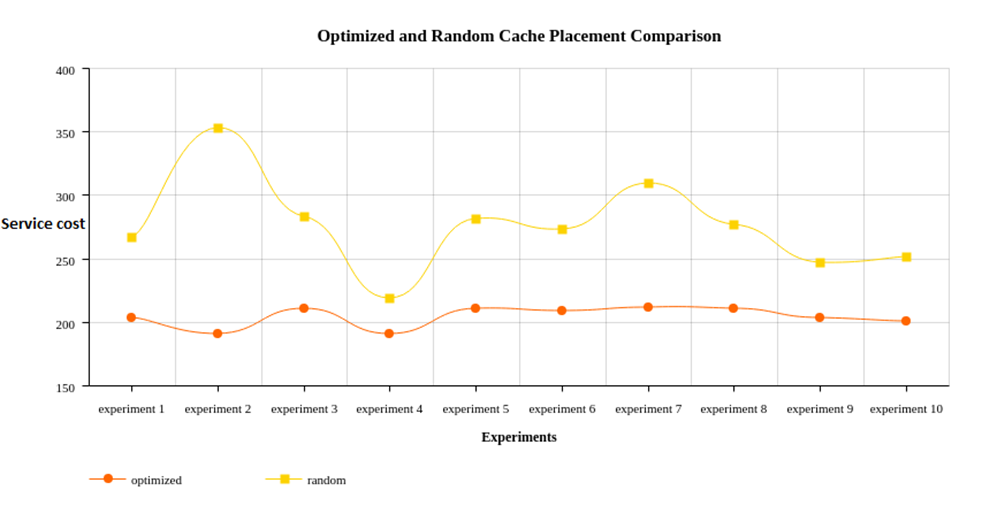


Figure 6. Optimized and random cache placement comparison for total serving times

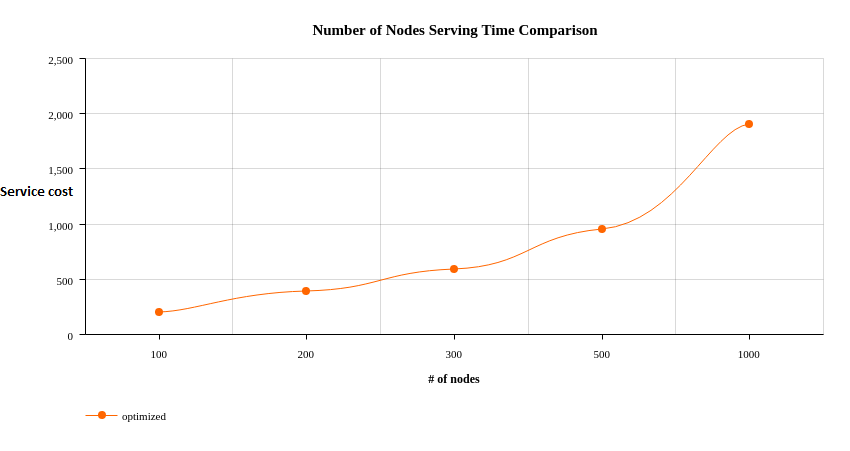


Figure 7. Serving time comparison for different number of nodes

harita, metin, iç mekan, kişi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure 8. Number of placed caches comparison for different number of nodes

GitHub link: <https://github.com/fyilmazz/uncapacitated-facility-location/tree/dev>