An Agent-based Home Exchange Model To Reduce Commuting Times of University Students

Yusuf Buyruk, Şehnaz Cenani, Gülen Çağdaş

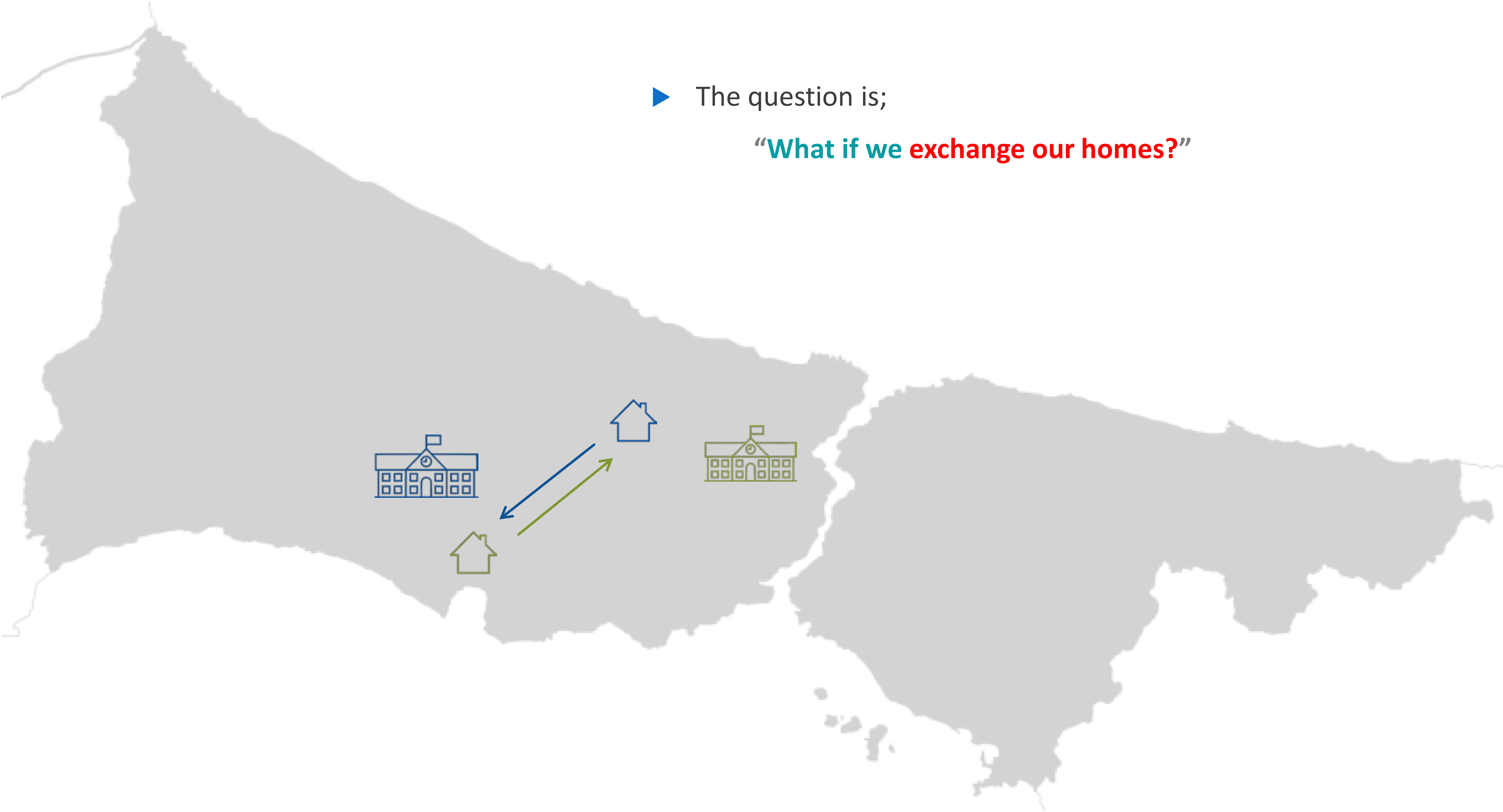
Istanbul Technical University
Graduate School of Science, Engineering & Technology
Department of Informatics
Architectural Design Computing PhD Program

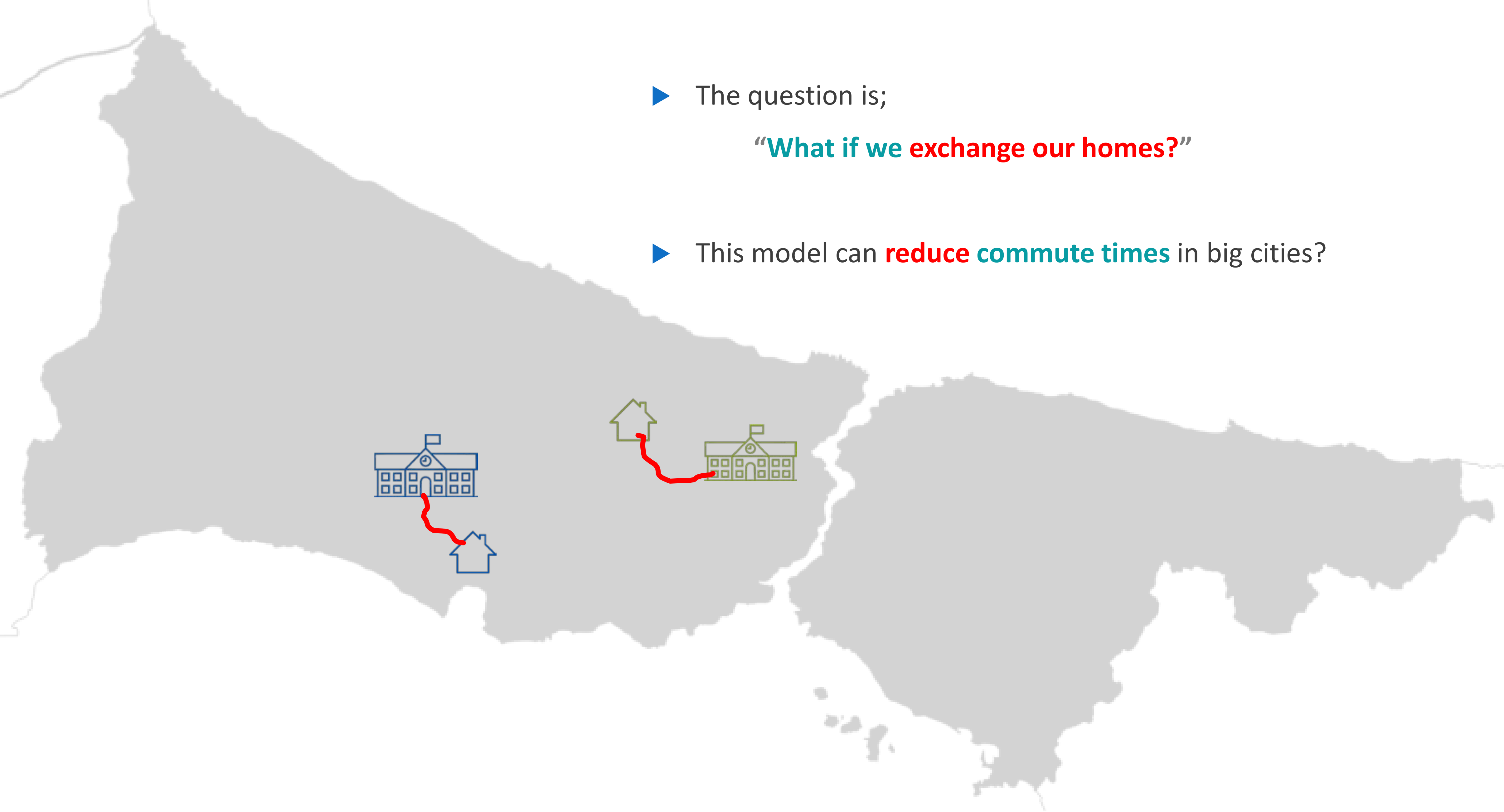
ISTANBUL, TURKEY

- People spend a considerable amount of their **time** **commuting to and from work or school.**

- The question is;

  “**What if we exchange our homes?**”





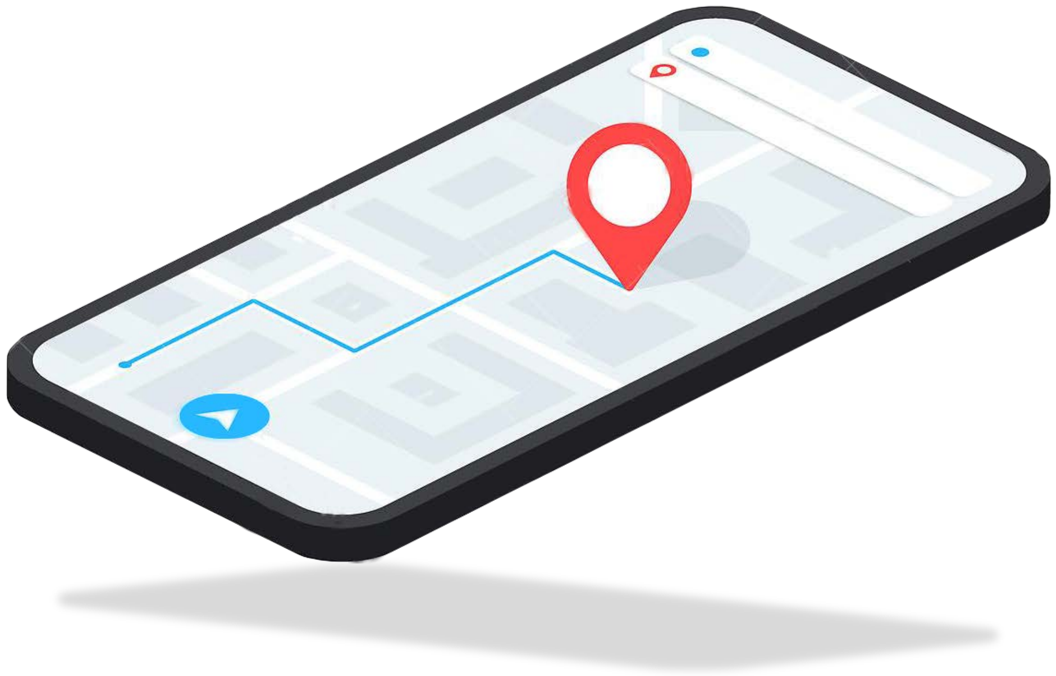
- The question is;

  “**What if we exchange our homes?**”

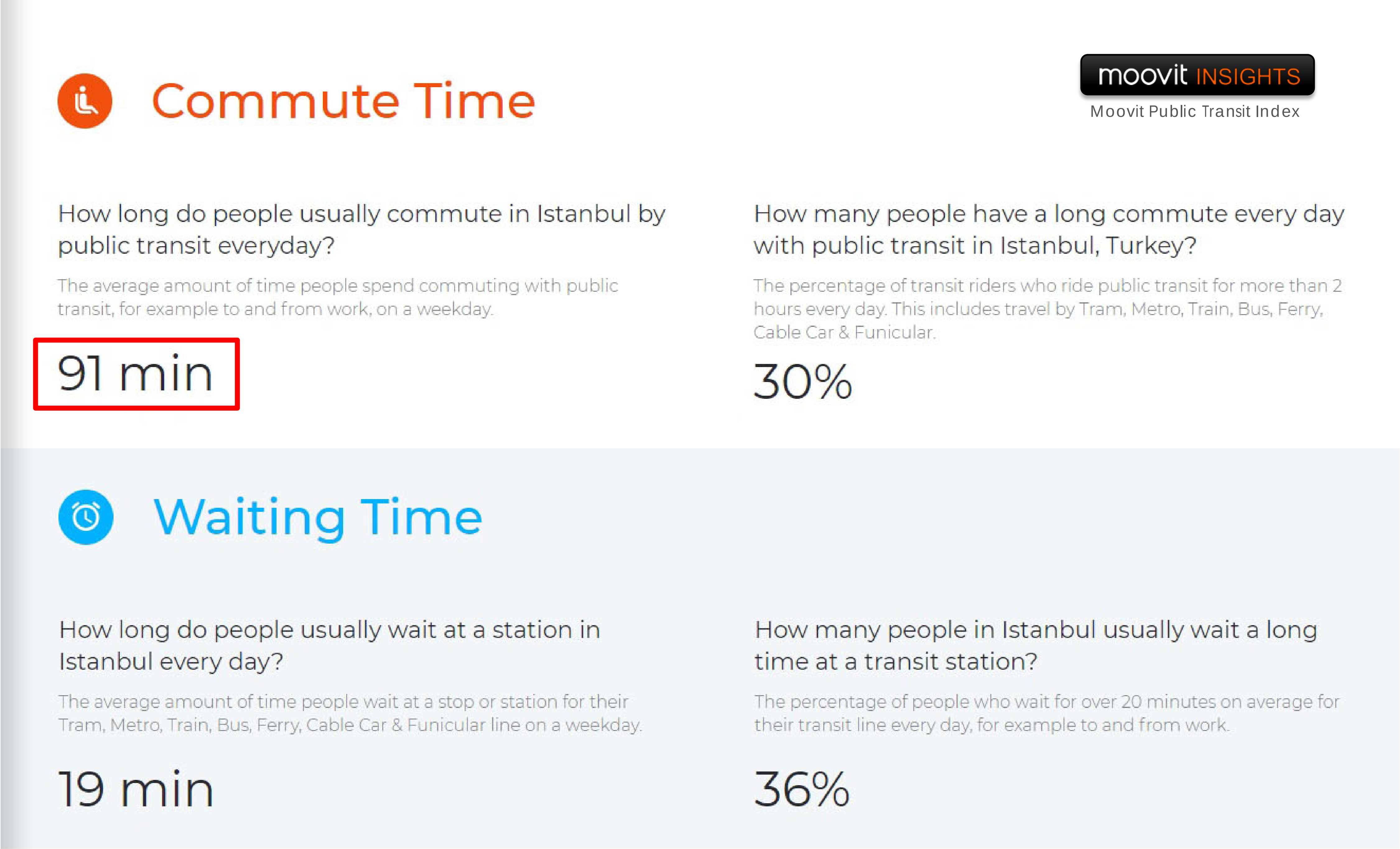
- This model can **reduce commute times** in big cities?

Home Exchange Model for University Students

- In order to contribute to developing smart and sustainable urban planning policies, this study proposes **home exchange model** for university students to **reduce commuting times**.
- This study aims at developing a computer application to improve the feasibility of smart and sustainable city concepts.
- In accordance with this purpose, **a problem specific matching algorithm** has been designed for home exchange problem and the algorithm has been applied on a virtual data.
- The results of the home exchange model are presented.



- MoovIt is a **mobile application** and a **web service** that **provides** travelers with **public transit data**, including the best route for their journey.
- **Users also can participate** in MoovIt Community and contribute with sending active reports about their travel experience, such as bus congestion levels, cleanliness, and more to help others have a better travel experience.
- **MoovIt analyzes usage data and travel patterns** and **compiles the MoovIt Public Transit Index** with statistics about the **average commute times** for different cities across the world.


moovit INSIGHTS
Moovit Public Transit Index

Commute Time

How long do people usually commute in Istanbul by public transit everyday?

The average amount of time people spend commuting with public transit, for example to and from work, on a weekday.

91 min

How many people have a long commute every day with public transit in Istanbul, Turkey?

The percentage of transit riders who ride public transit for more than 2 hours every day. This includes travel by Tram, Metro, Train, Bus, Ferry, Cable Car & Funicular.

30%

Waiting Time

How long do people usually wait at a station in Istanbul every day?

The average amount of time people wait at a stop or station for their Tram, Metro, Train, Bus, Ferry, Cable Car & Funicular line on a weekday.

19 min

How many people in Istanbul usually wait a long time at a transit station?

The percentage of people who wait for over 20 minutes on average for their transit line every day, for example to and from work.

36%

Figure 1. MoovIt Istanbul City Public Transit Index Report [1]

This study consists of **four** steps

1. creating home and university databases
2. constructing a travel time distance matrix
3. initial state distribution
4. home exchange matching

1

creating home and **university** databases

university database

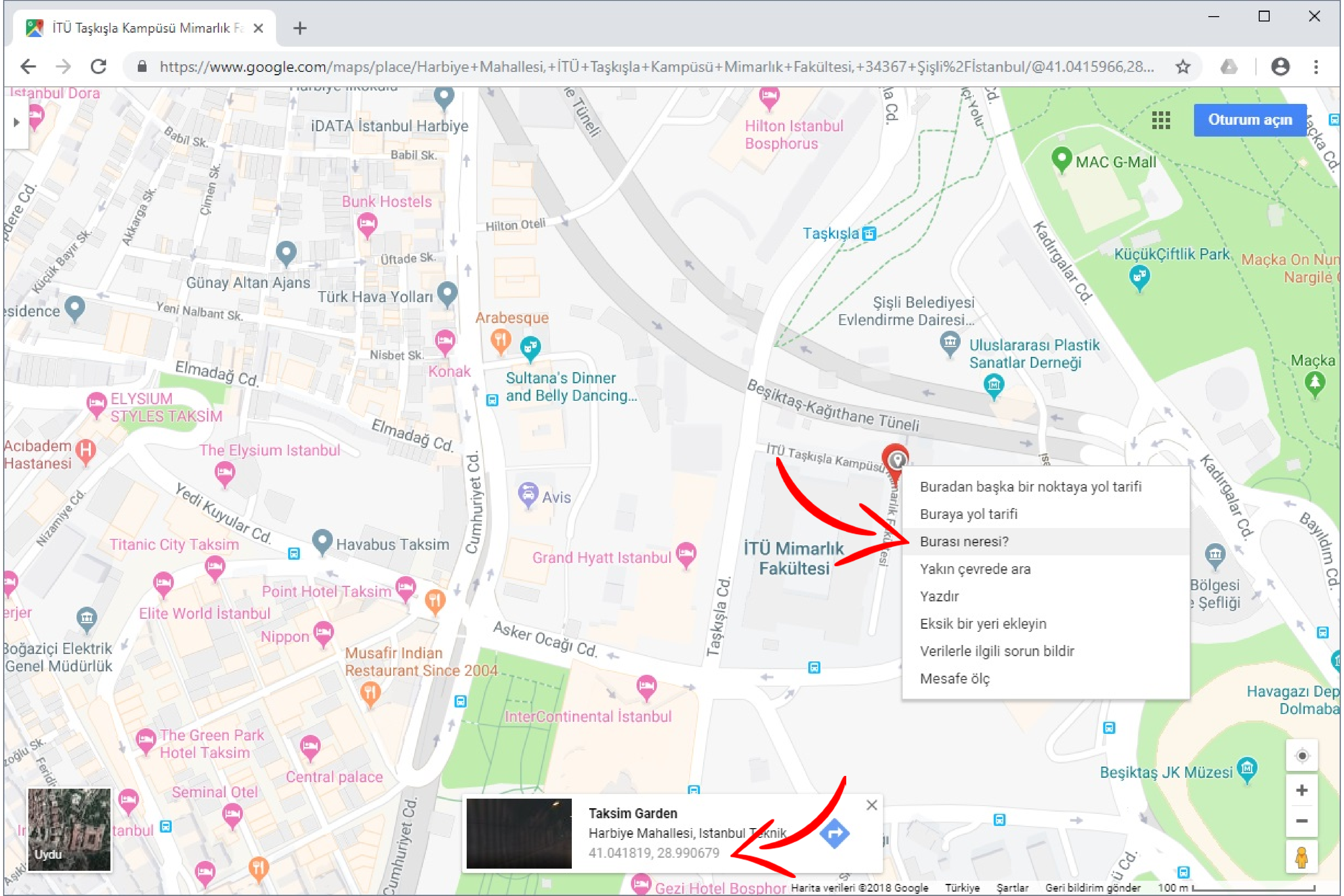
- **25 university campus locations**, which are located at European side of Istanbul city, have been chosen for university database.

1. Altınbaş Üniversitesi
2. Bahçeşehir Üniversitesi
3. Beykent Üniversitesi
4. Boğaziçi Üniversitesi
5. Galatasaray Üniversitesi
6. Haliç Üniversitesi
7. İstanbul Arel Üniversitesi
8. İstanbul Aydın Üniversitesi
9. İstanbul Bilgi Üniversitesi
10. İstanbul Bilim Üniversitesi
11. İstanbul Gelişim Üniversitesi
12. İstanbul Kültür Üniversitesi
13. İstanbul Rumeli Üniversitesi
14. İTÜ Ayazağa Yerleşkesi
15. İTÜ Taşkışla Yerleşkesi
16. İstanbul Üniversitesi Tıp Fakültesi
17. Cerrahpaşa Tıp Fakültesi
18. İstanbul Yeni Yüzyıl Üniversitesi
19. İstinye Üniversitesi
20. Kadir Has Üniversitesi
21. Koç Üniversitesi
22. MEF Üniversitesi
23. MSGSÜ Mimarlık Fakültesi
24. Nişantaşı Üniversitesi
25. Yıldız Teknik Üniversitesi

Table 1. University campus locations, which are located at European side of Istanbul city.

1

creating
home and
university
databases



1

creating **home** and university databases

home database

- For home database, rental house search sites have been browsed. The search filtered by number of rooms and rental price then **180 home locations** have been chosen.

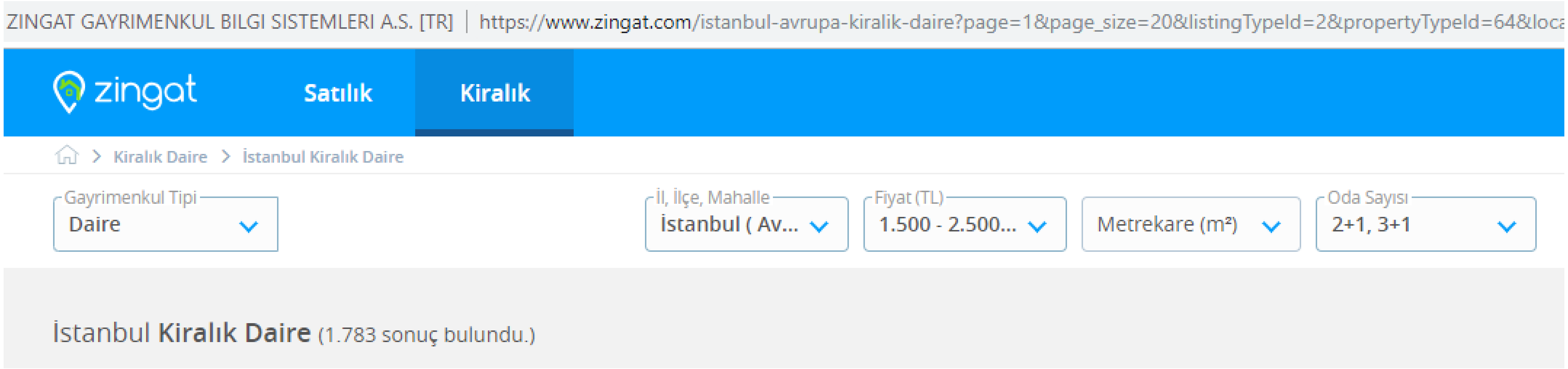
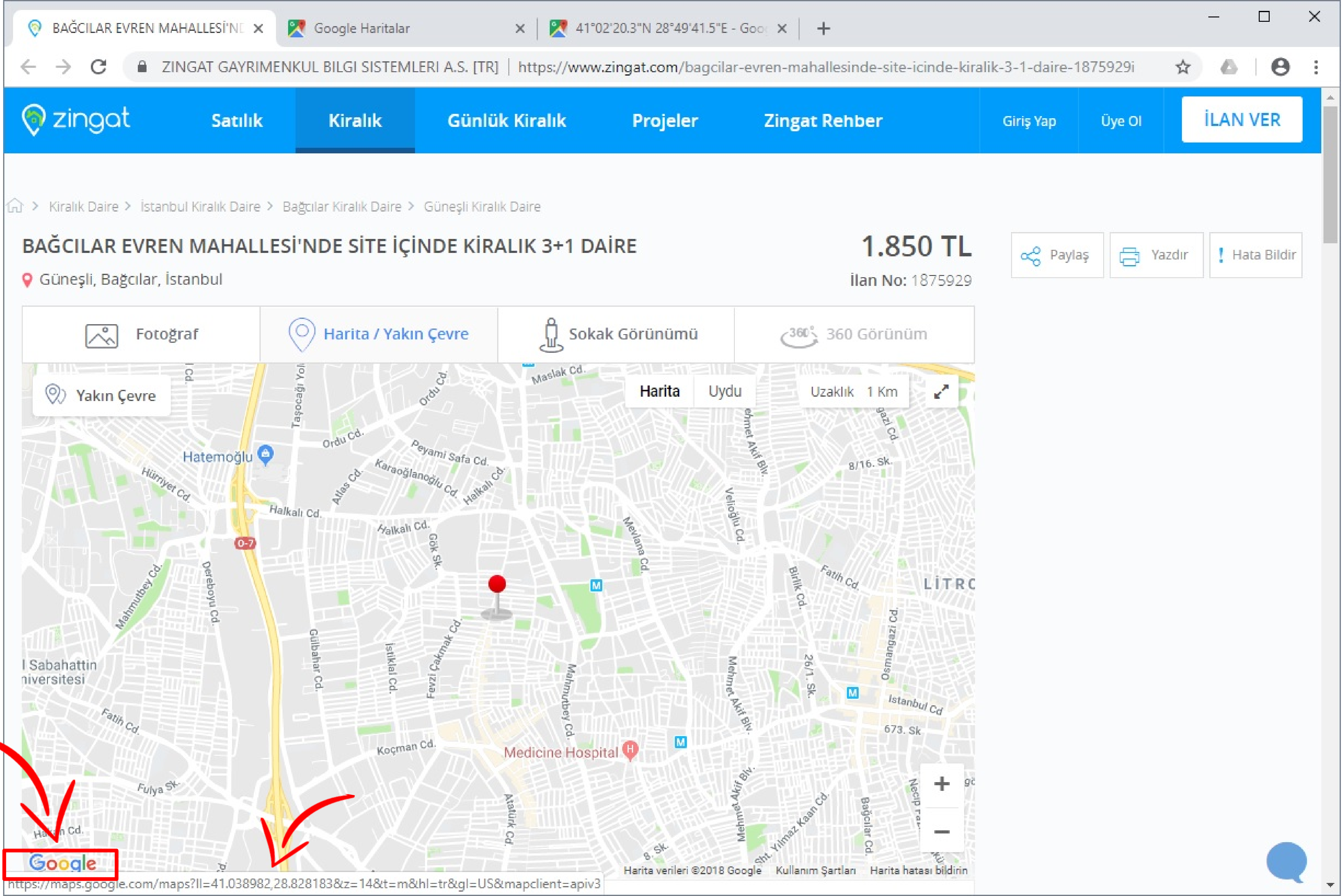


Figure 2. The search filtered by number of rooms (2+1 and 3+1), rental price (1500-2000) and location (Istanbul-European side).

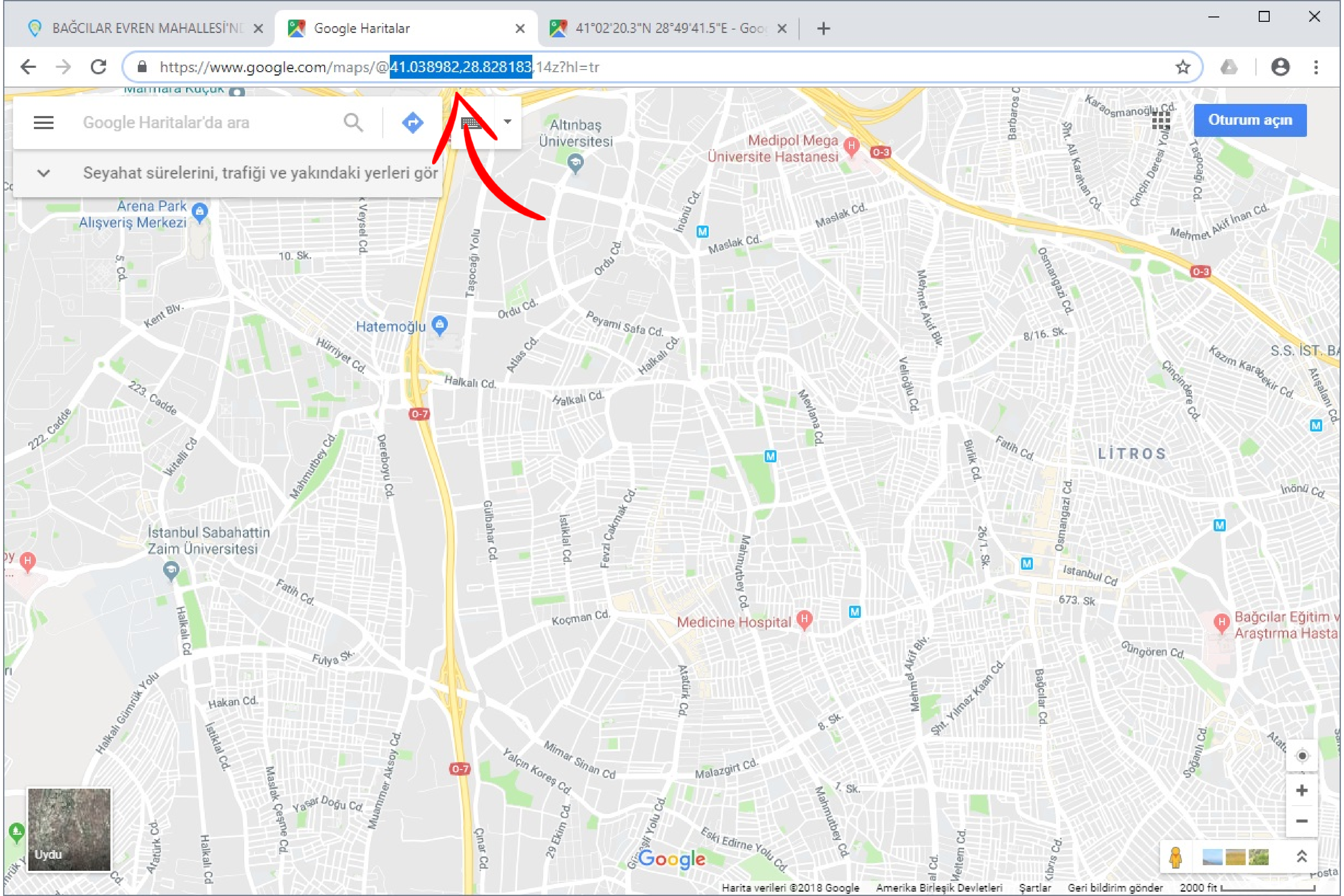
1

creating **home** and university databases



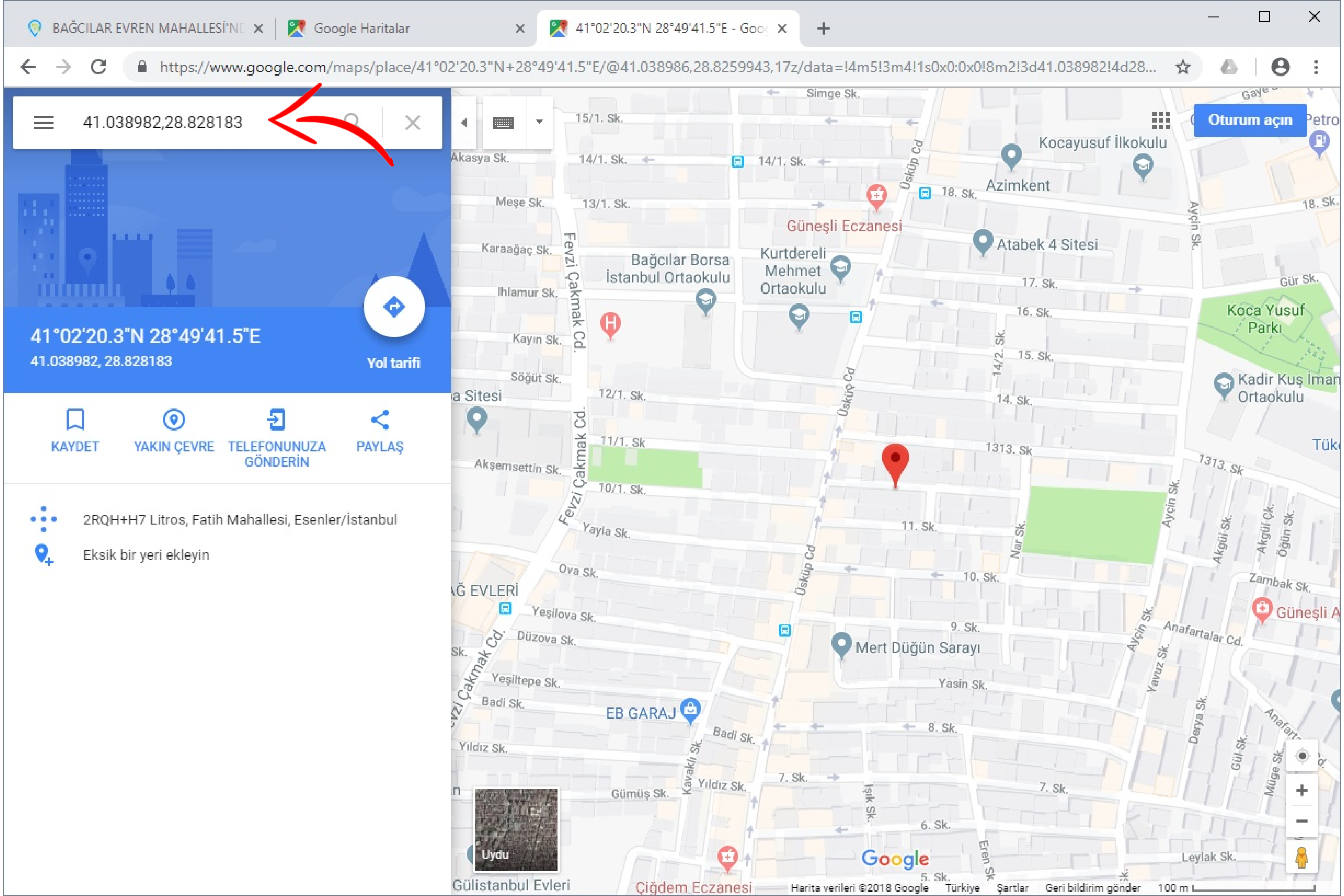
1

creating **home** and university databases



1

creating **home** and university databases



1

creating **home** and **university** databases

Konut Veritabanı 180.txt - Not Defteri

Dosya Düzen Biçim Görünüm Yardım

```
41.017275,28.621058
41.015626,28.641038
41.002118,28.64206
41.01041,28.658824
41.013632,28.659682
41.104785,28.669164
40.976901,28.674569
41.015713,28.682207
41.012953,28.68434
41.001278,28.7044
40.997326,28.706803
40.978818,28.715384
40.98939,28.717436
40.982883,28.718141
40.974487,28.72847
41.036703,28.76596
41.111333,28.768331
41.03424,28.77197
41.117742,28.772708
41.054337,28.773355
40.989602,28.777756
41.04739,28.77837
40.979266,28.779856
41.065337,28.780409
40.990265,28.78094
40.993374,28.782763
41.044663,28.783879
41.043424,28.79051
41.020506,28.791522
```

Üniversite Veritabanı 25.txt - Not Defteri

Dosya Düzen Biçim Görünüm Yardım

```
41.057462, 28.820425 Altınbaş Üniversitesi Mahmutbey, DİLmenler Cd. No. 26 Bağcılar, İstanbul
41.042474, 29.009247 Bahçeşehir Üniversitesi Çırağan Caddesi Osmanpaşa Mektebi Sokak No: 4-6 3453 Beşik
41.117575, 29.003741 Beykent Üniversitesi Ayazağa Mahallesi, Hadım Koru Yolu Caddesi, No:19 Sarıyer/İST
41.085028, 29.050936 Boğaziçi Üniversitesi Boğaziçi Üniversitesi 34342 Bebek/Istanbul Türkiye
41.046197, 29.019900 Galatasaray Ünivesitesi Galatasaray Üniversitesi Çırağan Cad. No:36 34349 Ortaköy/
41.056287, 28.950828 Haliç Üniversitesi Sütlüce Mahallesi, İmrahor Caddesi, No:11, Beyoğlu, istanbul
41.055397, 28.500257 İstanbul Arel Üniversitesi -Türkoba Mah. Erguvan Sokak No: 26/K Tepekent-B.Çekmece
40.992238, 28.797837 İstanbul Aydın Üniversitesi Florya Halit Aydın Kampüsü Beşyol Mah. İnönü Cad. No:
41.067305, 28.945449 İstanbul Bilgi Üniversitesi Eski Silahtarağa Elektrik Santralı Kazım Karabekir Cad
41.069329, 29.012314 İstanbul Bilim Üniversitesi Büyükdere Cad. No:120 Esentepe-Şişli İSTANBUL
40.998341, 28.698959 İstanbul Gelişim Üniversitesi Cihangir Mahallesi Şehit Jandarma Komando Er Hakan Ö
40.995333, 28.867355 İstanbul Kültür Üniversitesi Ataköy Yerleşkesi, 34156 Bakırköy / İSTANBUL
41.079226, 28.270118 İstanbul Rumeli Üniversitesi Yeni Mahalle Mah. Mehmet Silivrili Cad. No: 38/8 Post
41.105567, 29.023021 İstanbul Teknik Üniversitesi İTÜ Ayazağa Yerleşkesi yeni Rektörlük Binası Kat:2 MA
41.041266, 28.989501 İstanbul Teknik Üniversitesi Harbiye Mahallesi, İTÜ Taşkışla Kampüsü Mimarlık Fakü
41.015658, 28.932926 İstanbul Üniversitesi Tıp Fakültesi Topkapı Mahallesi, Turgut Özal Millet Cd, 3409
41.005466, 28.940293 İstanbul Üniversitesi Cerrahpaşa Tıp Fakültesi Kampüsü Cerrah Paşa Mahallesi, Cerr
41.016437, 28.906749 İstanbul Yeni Yüzyıl Üniversitesi Topkapı Dr. Azmi Ofluoğlu Yerleşkesi Maltepe Mah
41.014710, 28.905237 İstinye Üniversitesi Maltepe Mahallesi, 34010 Zeytinburnu/İstanbul
41.024739, 28.959062 Kadir Has Üniversitesi Cibali Mahallesi 34083 Fatih/İstanbul
41.212741, 29.087227 Koç Üniversitesi Mühendislik Fakültesi  Rumelifeneri Mahallesi, Rumelifeneri Yolu,
41.108889, 29.008604 MEF Üniversitesi Huzur Mh., Maslak Ayazağa Cd. No:4, 34396 Sarıyer/İstanbul
41.029705, 28.988940 Mimar Sinan Güzel Sanatlar Üniversitesi Mimarlık Fakültesi Pürtelaş Hasan Efendi M
41.070346, 28.961813 Nişantaşı Üniversitesi Merkez Mahallesi, Hasbahçe Cd. No:88, 34406 Kağıthane/İstan
41.052082, 29.010615 Yıldız Teknik Üniversitesi Yıldız Mh., 34349 Beşiktaş/İstanbul
```

Figure 6. Home and university location databases .

Distance Matrix API

2

constructing a travel time **distance matrix**

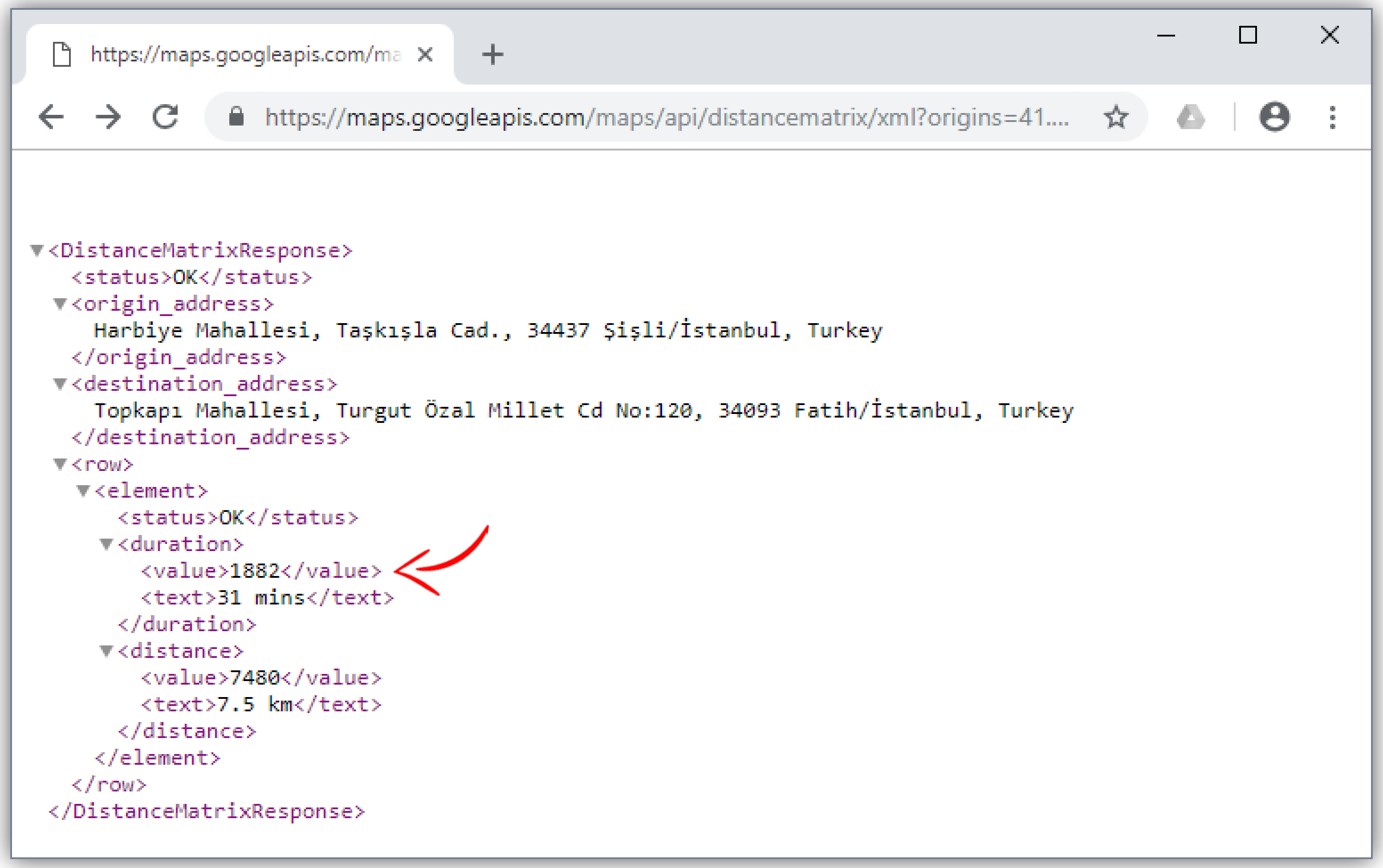
- In the second step, public transport travel times from homes to university campuses have been acquired from Distance Matrix API service.
- The Distance Matrix API is a service that provides **travel distance** and **time** for a matrix of origins and destinations [2].
- Distance Matrix API can be accessed through an HTTP interface, with requests constructed as a URL string, using **origins**, **destinations** and **optional parameters**, along with an **API key**.
- A Distance Matrix API request takes the following form:
  - https://maps.googleapis.com/maps/api/distancematrix/***outputFormat***?***parameters***

2

constructing a travel time **distance matrix**

sample request and response (XML)

https://maps.googleapis.com/maps/api/distancematrix/xml?origins=41.041266,28.989501&destinations=41.015658,28.932926
&units=metric&language=en-US&mode=transit&traffic_model=best_guess&key=YOUR_API_KEY




```xml
<DistanceMatrixResponse>
  <status>OK</status>
  <origin_address>
    Harbiye Mahallesi, Taşkışla Cad., 34437 Şişli/İstanbul, Turkey
  </origin_address>
  <destination_address>
    Topkapı Mahallesi, Turgut Özal Millet Cd No:120, 34093 Fatih/İstanbul, Turkey
  </destination_address>
  <row>
    <element>
      <status>OK</status>
      <duration>
        <value>1882</value>
        <text>31 mins</text>
      </duration>
      <distance>
        <value>7480</value>
        <text>7.5 km</text>
      </distance>
    </element>
  </row>
</DistanceMatrixResponse>
```


Figure 7. Distance Matrix API service sample XML request and response [2]

2

constructing
a travel time
**distance
matrix**

parameters

PARAMETRE	VALUES
outputFormat	*json | xml*
origins	*[latitude/longitude coordinates] | [address]*
destinations	
key	*[YOUR_API_KEY]*
mode	*driving | walking | bicycling | transit*
transit_mode	*bus | subway | train | tram | rail*
transit_routing_preferences	*less_walking | fewer_transfer*
traffic_model	*best_guess | pessimistic | optimistic*
departure_time	*UTC time*
arrival_time	
language	*en | tr | …*
region	
avoid	*tolls | highways | ferries | indoor*
units	*metric | imperial*

Table 2. Distance Matrix API service parameters [2]

2

constructing a travel time **distance matrix**

```csharp
static void Main(string[] args)
{

    List<String> destinations = new List<String>();
    destinations.Add("41.057462,28.820425"); // Altinbas
    destinations.Add("41.042474,29.009247"); // Bahcesehir
    destinations.Add("41.117575,29.003741"); // Beykent
    destinations.Add("41.085028,29.050936"); // Bogazici
    destinations.Add("41.045975,29.019902"); // Galatasaray
    destinations.Add("41.056287,28.950828"); // Halic
    destinations.Add("41.055397,28.500257"); // Arel
    destinations.Add("40.992238,28.797837"); // Aydin
    destinations.Add("41.067305,28.945449"); // Bilgi
    destinations.Add("41.069329,29.012314"); // Bilim
    destinations.Add("40.998341,28.698959"); // Gelisim
    destinations.Add("40.995333,28.867355"); // Kultur
    destinations.Add("41.079226,28.270118"); // Rumeli
    destinations.Add("41.105567,29.023021"); // ITU Ayazaga
    destinations.Add("41.041266,28.989501"); // ITU Taskisla
    destinations.Add("41.015658,28.932926"); // Istanbul Capa
    destinations.Add("41.005466,28.940293"); // Istanbul Cerrahpasa
    destinations.Add("41.016437,28.906749"); // Yeni Yuzyil
    destinations.Add("41.014710,28.905237"); // Istinye
    destinations.Add("41.024739,28.959062"); // Kadir Has
    destinations.Add("41.206130,29.075209"); // Koc
    destinations.Add("41.108889,29.008604"); // MEF
    destinations.Add("41.029705,28.988940"); // Mimar Sinan
    destinations.Add("41.070346,28.961813"); // Nisantasi
    destinations.Add("41.052082,29.010615"); // Yildiz Teknik
```

Program.cs — Line 12, Column 40 — Tab Size: 4 — C#

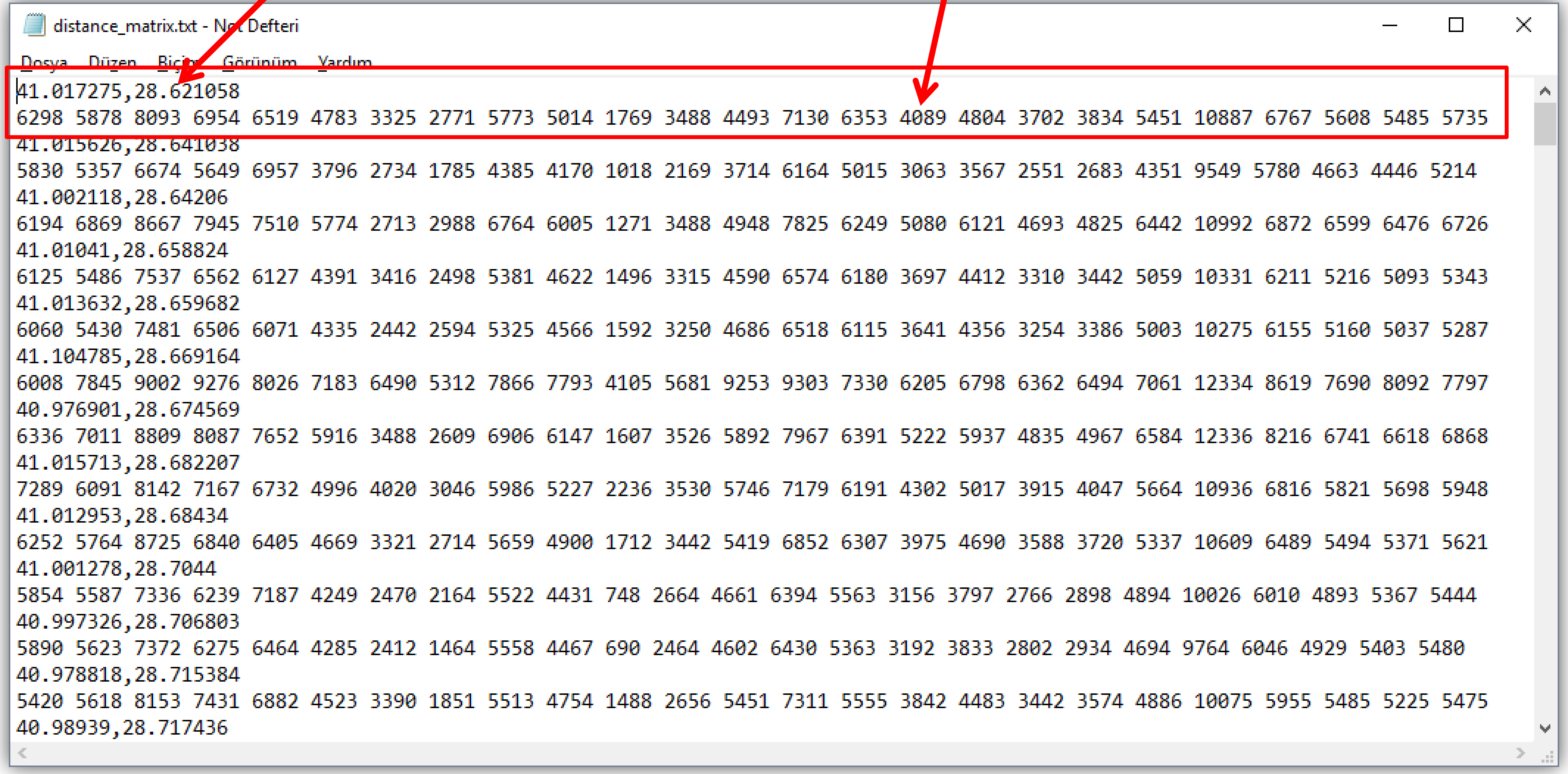
```csharp
StringBuilder sb = new StringBuilder();
using (StreamReader sr = new StreamReader("origins_database.txt"))
{
    String line = String.Empty;
    while ((line = sr.ReadLine()) != null)
    {
        sb.AppendLine(line);
        WebClient client = new WebClient();
        client.QueryString.Add("key", "YOUR_API_KEY"); // TODO: &key=YOU
        client.QueryString.Add("units", "metric");
        client.QueryString.Add("language", "tr");
        client.QueryString.Add("mode", "transit"); // transit_mode | tra
        client.QueryString.Add("traffic_model", "best_guess"); // traffi
        client.QueryString.Add("destinations", String.Join("|", destinat
        client.QueryString.Add("origins", line);

        Stream response = client.OpenRead("https://maps.googleapis.com/m
        XmlDocument xmlDoc = new XmlDocument();
        xmlDoc.Load(response);
        XmlNodeList durations = xmlDoc.SelectNodes("//duration/value");

        foreach (XmlNode duration in durations)
            sb.Append(String.Format("{0} ", duration.InnerText));
        sb.AppendLine();
    }
}

using (StreamWriter sw = new StreamWriter("distance_matrix.txt", append:
    sw.WriteLine(sb.ToString());
```

Program.cs — Line 42, Column 52 — Tab Size: 4 — C#

Figure 8. Distance matrix C# code, screenshots

home location
GPS coordinates

duration values to commute
university locations




```
41.017275,28.621058
6298 5878 8093 6954 6519 4783 3325 2771 5773 5014 1769 3488 4493 7130 6353 4089 4804 3702 3834 5451 10887 6767 5608 5485 5735
41.015626,28.641038
5830 5357 6674 5649 6957 3796 2734 1785 4385 4170 1018 2169 3714 6164 5015 3063 3567 2551 2683 4351 9549 5780 4663 4446 5214
41.002118,28.64206
6194 6869 8667 7945 7510 5774 2713 2988 6764 6005 1271 3488 4948 7825 6249 5080 6121 4693 4825 6442 10992 6872 6599 6476 6726
41.01041,28.658824
6125 5486 7537 6562 6127 4391 3416 2498 5381 4622 1496 3315 4590 6574 6180 3697 4412 3310 3442 5059 10331 6211 5216 5093 5343
41.013632,28.659682
6060 5430 7481 6506 6071 4335 2442 2594 5325 4566 1592 3250 4686 6518 6115 3641 4356 3254 3386 5003 10275 6155 5160 5037 5287
41.104785,28.669164
6008 7845 9002 9276 8026 7183 6490 5312 7866 7793 4105 5681 9253 9303 7330 6205 6798 6362 6494 7061 12334 8619 7690 8092 7797
40.976901,28.674569
6336 7011 8809 8087 7652 5916 3488 2609 6906 6147 1607 3526 5892 7967 6391 5222 5937 4835 4967 6584 12336 8216 6741 6618 6868
41.015713,28.682207
7289 6091 8142 7167 6732 4996 4020 3046 5986 5227 2236 3530 5746 7179 6191 4302 5017 3915 4047 5664 10936 6816 5821 5698 5948
41.012953,28.68434
6252 5764 8725 6840 6405 4669 3321 2714 5659 4900 1712 3442 5419 6852 6307 3975 4690 3588 3720 5337 10609 6489 5494 5371 5621
41.001278,28.7044
5854 5587 7336 6239 7187 4249 2470 2164 5522 4431 748 2664 4661 6394 5563 3156 3797 2766 2898 4894 10026 6010 4893 5367 5444
40.997326,28.706803
5890 5623 7372 6275 6464 4285 2412 1464 5558 4467 690 2464 4602 6430 5363 3192 3833 2802 2934 4694 9764 6046 4929 5403 5480
40.978818,28.715384
5420 5618 8153 7431 6882 4523 3390 1851 5513 4754 1488 2656 5451 7311 5555 3842 4483 3442 3574 4886 10075 5955 5485 5225 5475
40.98939,28.717436
```

Figure 9. Distance Matrix output

2

constructing
a travel time
distance
matrix

3

initial state distribution

initial state distribution

- In the third step, the initial state distribution has been created in order **to simulate real life** situation.

- In order to make real and virtual data closer, it is also tried to make the gap between average commuting time of virtual data and average commuting time in real life closer.

- According to Istanbul city public transit index report released by MoovIt public transport application,
  - the daily average amount of time that people spend commuting roundtrip with public transit on a weekday is **91 minutes** and
  - the daily average amount of time that people spend waiting at a stop or station on a weekday is **19 minutes.** [1]

3

initial state distribution

tournament selection

- **Tournament selection** method has been preferred to create initial state distribution.
- Tournament selection is a method of selecting an individual from a population of individuals in a population-based search algorithm.
- In this method, tournament size (k) number of individuals are chosen from the population at random then the best among them is selected.
- **Selection pressure** can be controlled by the parameter of **tournament size**.

3

initial state distribution

tournament selection

- tournament size (k) = 4
- 4 home locations are chosen randomly

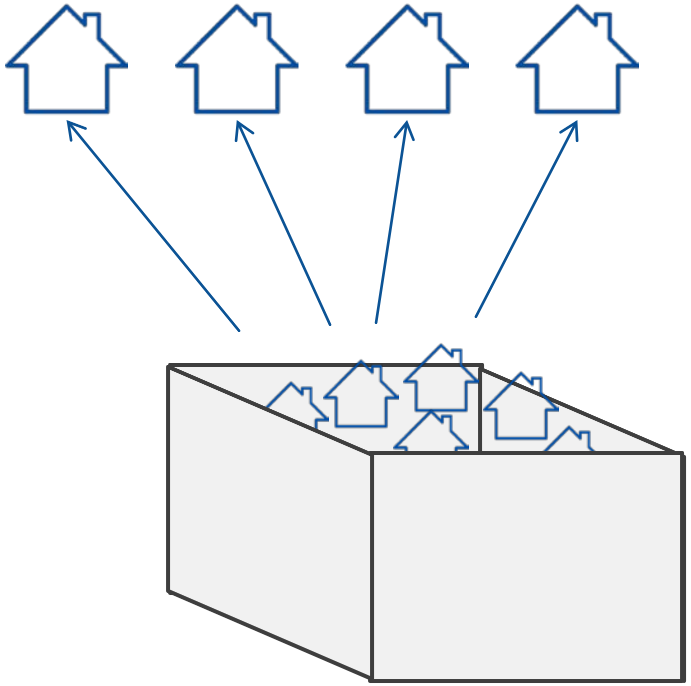


Figure 10a. Tournament selection

tournament selection

survival of the fittest

- the closest among them is selected and paired.

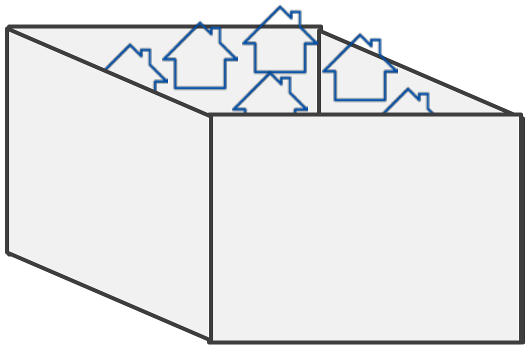


Figure 10b. Tournament selection

3

initial state
distribution

tournament selection

- This iteration is repeated until **each university** is paired with **6 homes.**
- **150 home** locations are selected and paired with **25 university** locations

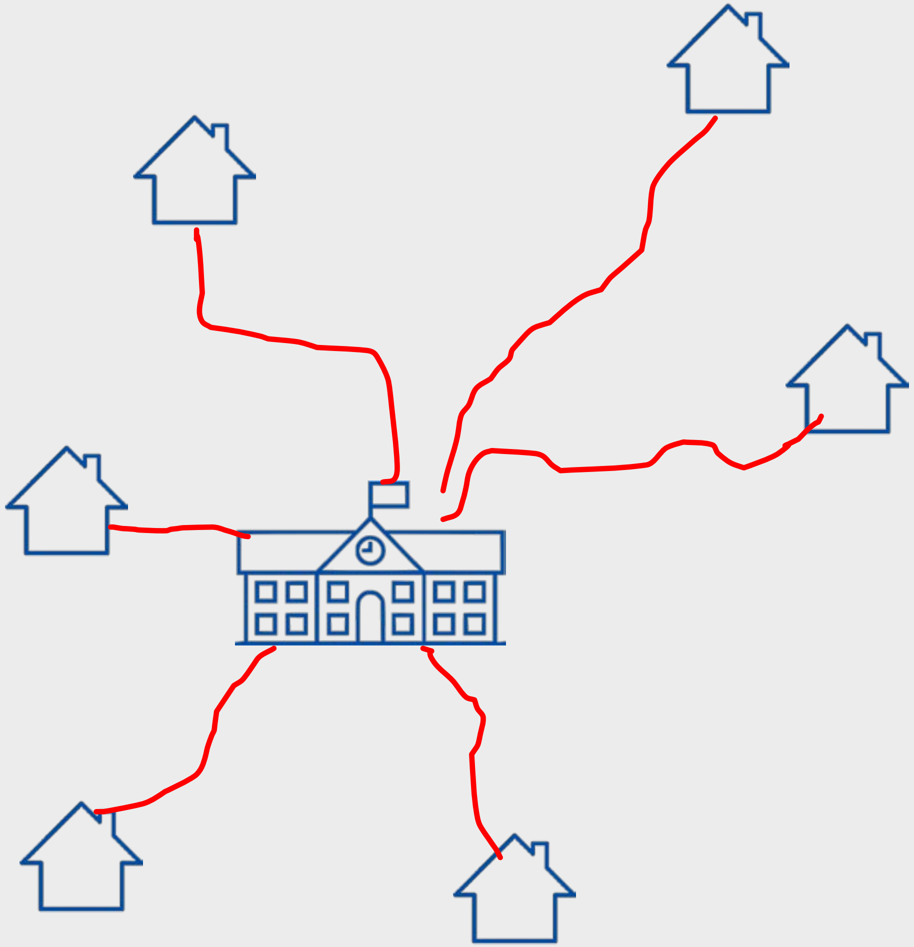


Figure 10c. Tournament selection

3

initial state
distribution

tournament selection

- Finally, **150 of 180** homes in home database are selected for initial state distribution.
- **30** homes are not selected.
- **Extreme values** are eliminated.
- Initial state distribution can simulate real life much closer.

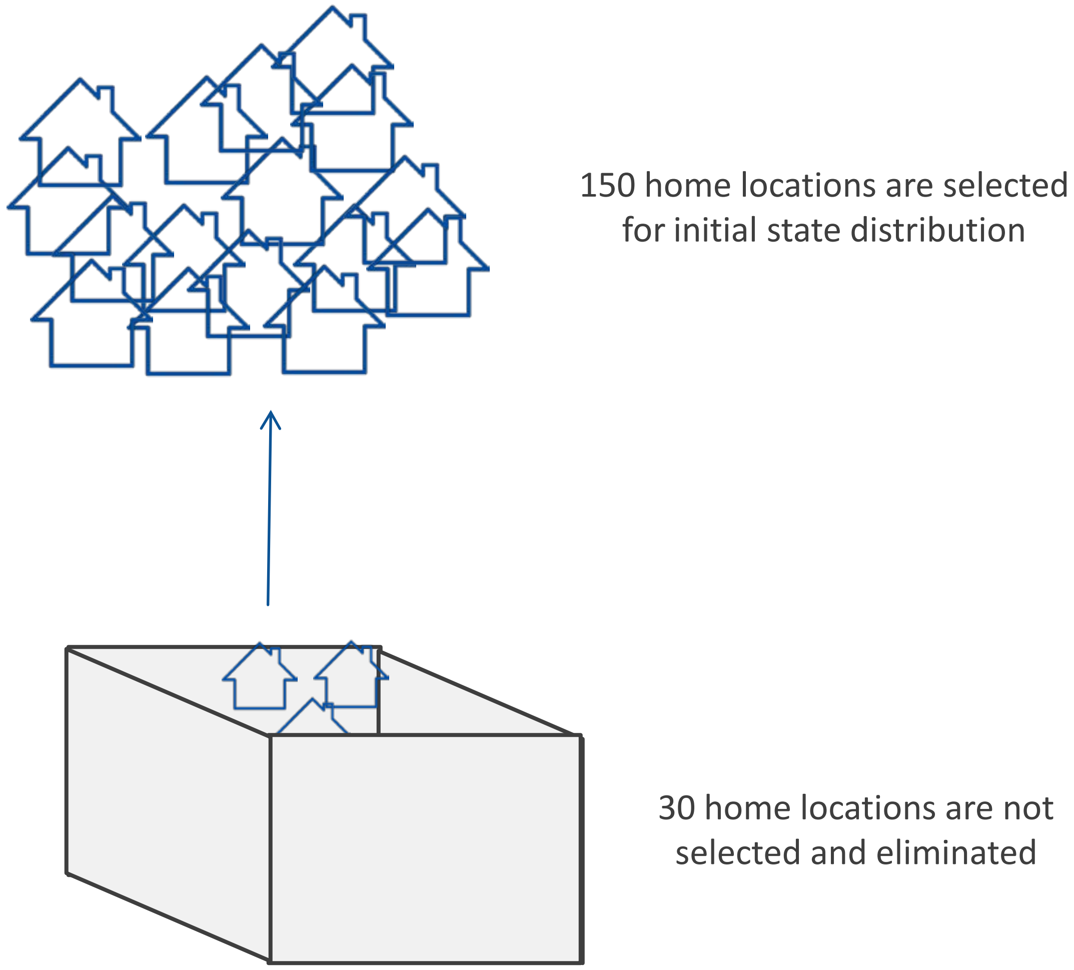


Figure 10d. Tournament selection

3

initial state distribution

tournament selection

- tournament size parameter = **4**
- At the end, average roundtrip commuting time of initial state distribution is calculated as **88 minutes**.
- According to Istanbul city public transit index report released by MoovIt public transport application,
  - the daily average amount of time that people spend commuting roundtrip with public transit on a weekday is **91 minutes**

tournament selection

```csharp
// TOURNAMENT SELECTION
for (int college_index = 0; college_index < 25; college_index++)
{
    for (int i = 0; i < 6; i++)
    {
        Agent currentBest = agents_loaded[rnd.Next(0, agents_loaded.Count)];

        for (int k = 1; k < 4; k++) // Tournament Size K = 4
        {
            Agent agent = agents_loaded[rnd.Next(0, agents_loaded.Count)];
            if (agent.Values[college_index] < currentBest.Values[college_index])
                currentBest = agent;
        }

        currentBest.Index = college_index;
        agents_loaded.Remove(currentBest);
        agents.Add(currentBest);
    }
}
```

Figure 11. Tournament selection C# code, screenshots

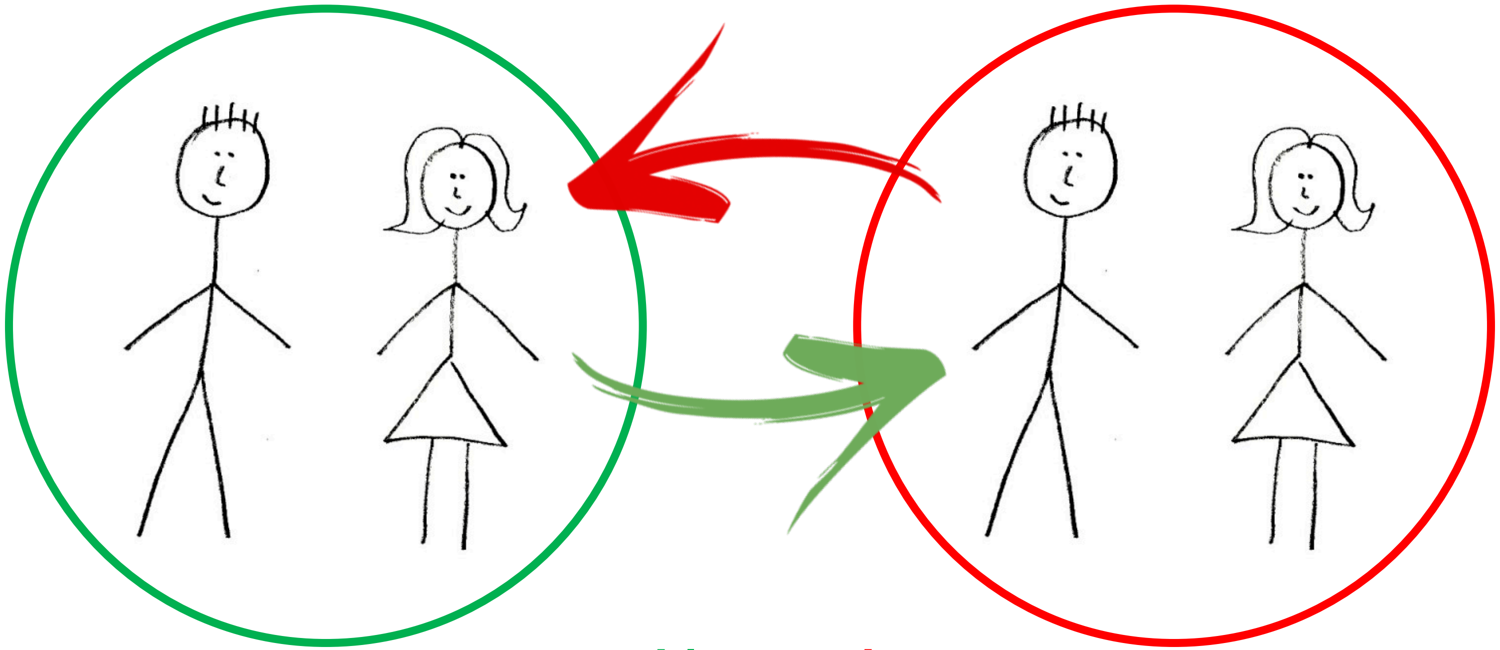
matching theory

- matching, allocation and exchange of discrete resources [3]
- In 1962 David Gale and Lloyd Shapley published one of the most influential papers in matching theory starting the literature in matching theory [3][4].
- This study introduces;
  - suitable solution concept called **stability** and
  - **two-sided matching** model
  - **deferred acceptance** algorithm

4

home
exchange
matching

stability



unstable match

- In the stable matching problem, if there is a man and a woman who would **prefer to be matched** with each other **more than their current matches**

Figure 12. Stability, unstable match

one-sided and **two-sided** matching

- **two-sided matching** model
  - two sides, such as firms and workers, students and schools, or men and women, that need to be matched with each other.
  - **stable marriage** problem

- **one-sided matching** model
  - matching agents are in the same set
  - **stable roommates** and **housing market** problem

one-sided and **two-sided** matching

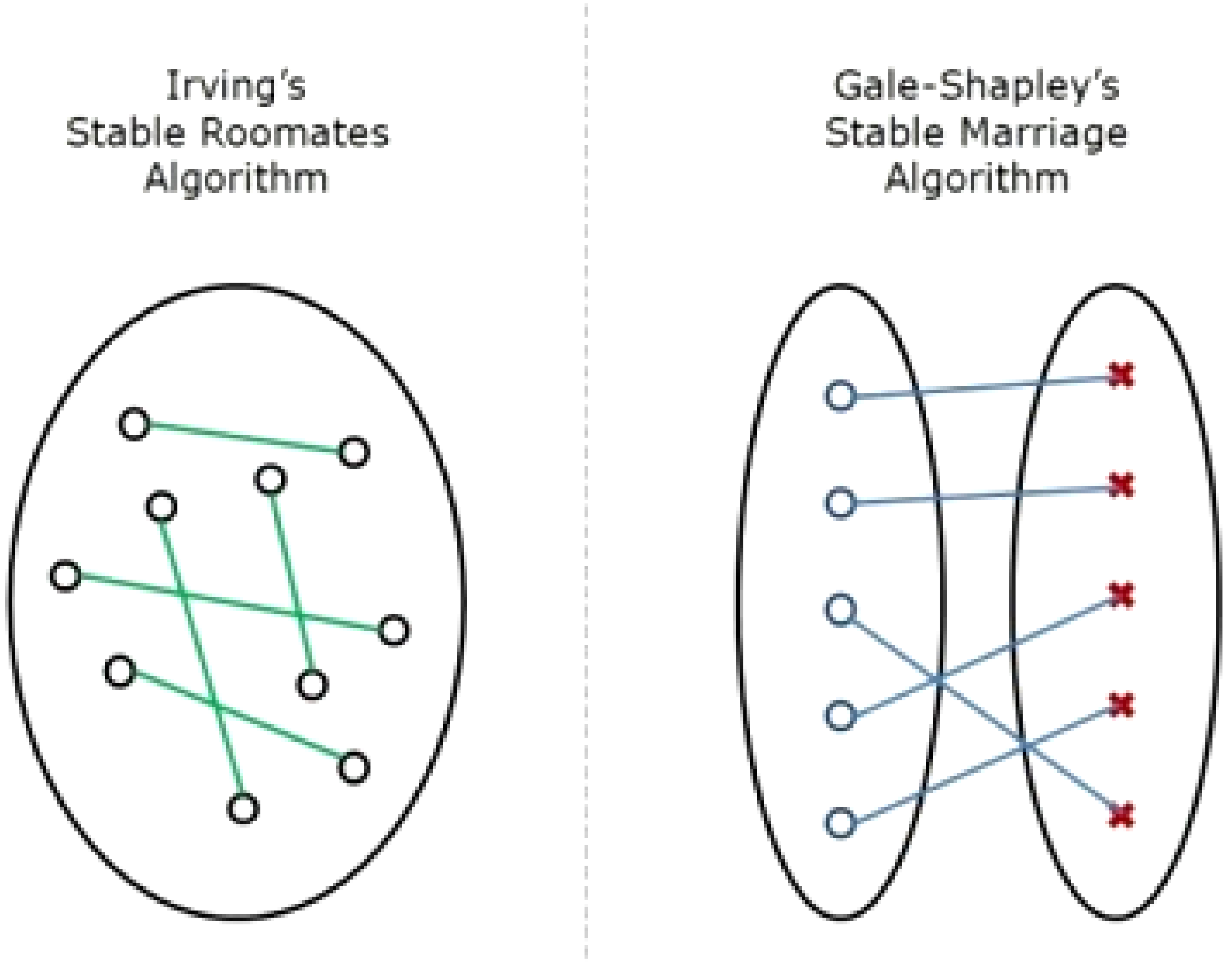


Figure 13. One-sided and two-sided matching [3]

4

home exchange matching

stable marriage matching algorithm

- The Gale-Shapley Algorithm to compute the stable matching for two-sided markets, such as the **stable marriage** problem and **the college-admissions** problem
  - **two-sided** matching algorithm [4]

- To solve **home exchange** problem, **one-sided** matching algorithm is need.

matching algorithms for **home exchange** problem

- Irving's Algorithm [5]
  - **stable roommates** problem
- Gale's the top trading cycle algorithm [6]
  - **housing market** problem
- Proposed Algorithm
  - **home exchange matching** problem

4

home exchange matching

stable roommates matching algorithm

- Irving's Algorithm to compute the stable matching for one-sided matchings such as the **stable roommates** problem [5]
  - **one-sided** matching algorithm
  - **stability**

- each agent has a preference list of **all other agents** and **all agents** are matched

- In home-exchange problem, each agent has a preference list of **only the agents** such that **a match reduces travel times** for both sides.

Stable Roommate

Stage 1

- Everybody proposes to their favorite. Order does not matter.
- Proposal recipients then pick their most best proposer and reject the rest.
- Those who got rejected keep proposing until accepted.
- If someone gets rejected by everyone else then no stable matching exists.

Stable Roommate

makes a proposal

accepts a proposal

rejects

\+ -

A	B	D	F	C	E
B	D	E	F	A	C
C	D	E	F	A	B
D	F	C	A	E	B
E	F	C	D	B	A
F	A	B	D	C	E

Table 3. Stable Roommate Algorithm workflow process

4

home exchange matching

Stable Roommate

makes a proposal

accepts a proposal

rejects

+ ↔ -

A	B	D	F	C	E
B	D	E	F	A	C
C	D	E	F	A	B
D	F	C	A	E	B
E	F	C	D	B	A
F	A	B	D	C	E

Table 3. Stable Roommate Algorithm workflow process

Stable Roommate

makes a proposal

accepts a proposal

rejects

+ ⟷ -

A	B	D	F	C	E
B	D	E	F	A	C
C	D	E	F	A	B
D	F	C	A	E	B
E	F	C	D	B	A
F	A	B	D	C	E

Table 3. Stable Roommate Algorithm workflow process

4

home exchange matching

Stable Roommate

makes a proposal

accepts a proposal

rejects

\+ ⟷ -

A	B	D	F	C	E
B	D	E	F	A	C
C	D	E	F	A	B
D	F	C	A	E	B
E	F	C	D	B	A
F	A	B	D	C	E

Table 3. Stable Roommate Algorithm workflow process

4

home exchange matching

Stable Roommate

	+				-
A	$\boxed{B}$	D	F	C	E
B	$\boxed{\cancel{D}}$	E	F	A	C
C	$\boxed{D}$	E	F	A	B
D	F	C	A	E	~~B~~
E	F	C	D	B	A
F	A	B	D	C	E

makes a proposal

accepts a proposal

rejects

Table 3. Stable Roommate Algorithm workflow process

Stable Roommate

□ makes a proposal

○ accepts a proposal

✕ rejects

+ ⟷ -

A	B	D	F	C	E
B		E	F	A	C
C	D	E	F	A	B
D	F	C	A	E	
E	F	C	D	B	A
F	A	B	D	C	E

Table 3. Stable Roommate Algorithm workflow process

4

home exchange matching

Stable Roommate

makes a proposal

accepts a proposal

rejects

\+ ⟷ -

A	B	D	F	C	E
B		E	F	A	C
C	D	E	F	A	B
D	F	C	A	E	
E	F	C	D	B	A
F	A	B	D	C	E

Table 3. Stable Roommate Algorithm workflow process

4

home exchange matching

Stable Roommate

makes a proposal

accepts a proposal

rejects

\+ ⟷ -

A	B	D	F	C	E
B		E	F	A	C
C	D	E	F	A	B
D	F	C	A	E	
E	F	C	D	B	A
F	A	B	D	C	E

Table 3. Stable Roommate Algorithm workflow process

Stable Roommate

makes a proposal

accepts a proposal

rejects

+ -

A	B	D	F	C	E
B		E	F	A	C
C	D	E	F	A	B
D	F	C	A	E	
E	~~F~~	C	D	B	A
F	A	B	D	C	~~E~~

Table 3. Stable Roommate Algorithm workflow process

Stable Roommate

4

home exchange matching

makes a proposal

accepts a proposal

rejects

+ ←→ -

A	B	D	F	C	E
B		E	F	A	C
C	D	E	F	A	B
D	F	C	A	E	
E		C	D	B	A
F	A	B	D	C	

Table 3. Stable Roommate Algorithm workflow process

4

home exchange matching

Stable Roommate

makes a proposal

accepts a proposal

rejects

\+ ⟷ -

A	B	D	F	C	E
B		E	F	A	C
C	D	E	F	A	B
D	F	C	A	E	
E		C	D	B	A
F	A	B	D	C	

Table 3. Stable Roommate Algorithm workflow process

4

home exchange matching

Stable Roommate

□ makes a proposal

○ accepts a proposal

✕ rejects

+ ⟷ -

A	B	D	F	C	E
B		E	F	A	C
C	D	E	F	A	B
D	F	C	A	E	
E		C	D	B	A
F	A	B	D	C	

Table 3. Stable Roommate Algorithm workflow process

Stable Roommate

Stage 2

- Everyone rejects those potential partners less desirable than their current accepted one.

4

home
exchange
matching

Stable Roommate

- makes a proposal
- accepts a proposal
- rejects

+ -

A	B	D	F	C	E
B		E	F	A	C
C	D	E	F	A	B
D	F	C	A	E	
E		C	D	B	A
F	A	B	D	C	

Table 3. Stable Roommate Algorithm workflow process

4

home
exchange
matching

Stable Roommate

makes a proposal

accepts a proposal

rejects

\+ ⟷ -

A	B	D	F	~~C~~	E
B		E	F	A	C
C	D	E	F	~~A~~	B
D	F	C	A	E	
E		C	D	B	A
F	A	B	D	C	

Table 3. Stable Roommate Algorithm workflow process

4

home exchange matching

Stable Roommate

makes a proposal

accepts a proposal

rejects

\+ ⟷ -

A	B	D	F		E
B		E	F	A	C
C	D	E	F		B
D	F	C	A	E	
E		C	D	B	A
F	A	B	D	C	

Table 3. Stable Roommate Algorithm workflow process

Stable Roommate

4

home exchange matching

makes a proposal

accepts a proposal

rejects

\+ ↔ -

A	B	D	F		~~E~~
B		E	F	A	C
C	D	E	F		B
D	F	C	A	E	
E		C	D	B	~~A~~
F	A	B	D	C	

Table 3. Stable Roommate Algorithm workflow process

Stable Roommate

□ makes a proposal

○ accepts a proposal

✕ rejects

\+ ⟷ -

A	B	D	F		
B		E	F	A	C
C	D	E	F		B
D	F	C	A	E	
E		C	D	B	
F	A	B	D	C	

Table 3. Stable Roommate Algorithm workflow process

Stable Roommate

makes a proposal

accepts a proposal

rejects

\+ ⟷ -

A	B	D	F		
B		E	F	A	~~C~~
C	D	E	F		~~B~~
D	F	C	A	E	
E		C	D	B	
F	A	B	D	C	

Table 3. Stable Roommate Algorithm workflow process

4

home exchange matching

Stable Roommate

makes a proposal

accepts a proposal

rejects

+ ⟷ -

A	B	D	F	
B		E	F	A
C	D	E	F	
D	F	C	A	E
E		C	D	B
F	A	B	D	C

Table 3. Stable Roommate Algorithm workflow process

Stable Roommate

□	makes a proposal
○	accepts a proposal
✕	rejects

\+ ⟷ -

A	B	D	F	
B		E	F	A
C	D	E	~~F~~	
D	F	C	A	E
E		C	D	B
F	A	B	D	~~C~~

Table 3. Stable Roommate Algorithm workflow process

Stable Roommate

□ makes a proposal

○ accepts a proposal

✕ rejects

\+ ⟷ -

A	B	D	F	
B		E	F	A
C	D	E		
D	F	C	A	E
E		C	D	B
F	A	B	D	

Table 3. Stable Roommate Algorithm workflow process

4

home exchange matching

Stable Roommate

makes a proposal

accepts a proposal

rejects

\+ -

A	B	~~D~~	F	
B		E	F	A
C	D	E		
D	F	C	~~A~~	E
E		C	D	B
F	A	B	D	

Table 3. Stable Roommate Algorithm workflow process

Stable Roommate

makes a proposal

accepts a proposal

rejects

\+ ⟷ -

A	B		F	
B		E	F	A
C	D	E		
D	F	C		E
E		C	D	B
F	A	B	D	

Table 3. Stable Roommate Algorithm workflow process

4

home exchange matching

Stable Roommate

makes a proposal

accepts a proposal

rejects

+ ⟷ -

A	B		F	
B		E	F	A
C	D	E		
D	F	C		~~E~~
E		C	~~D~~	B
F	A	B	D	

Table 3. Stable Roommate Algorithm workflow process

Stable Roommate

makes a proposal

accepts a proposal

rejects

\+ ⟷ -

A	B		F	
B		E	F	A
C	D	E		
D	F	C		
E		C		B
F	A	B	D	

Table 3. Stable Roommate Algorithm workflow process

Stable Roommate

Stage 3

- Find a participant who has more than one choice.
- Write their second preference X, then the last preference, Y of X.
- Repeat previous step until the starting player appears again.
- Every second preference and last preference then reject symmetrically.
- Do this until everybody has only one option.

4

home exchange matching

Stable Roommate

A

A has more than one choice

\+ ⟷ -

A	B		F	
B		E	F	A
C	D	E		
D	F	C		
E		C		B
F	A	B	D	

Table 3. Stable Roommate Algorithm workflow process

Stable Roommate

A

F

A has more than one choice

F is the <u>second</u> choice of A

+ -

A	B		F	
B		E	F	A
C	D	E		
D	F	C		
E		C		B
F	A	B	D	

Table 3. Stable Roommate Algorithm workflow process

4

home exchange matching

Stable Roommate

A D

F

A has more than one choice

F is the second choice of A

D is the last choice of F

	+			-
A	B		F	
B		E	F	A
C	D	E		
D	F	C		
E		C		B
F	A	B	D	

Table 3. Stable Roommate Algorithm workflow process

4

home exchange matching

Stable Roommate

A D

F C

A has more than one choice

F is the <u>second</u> choice of A

D is the <u>last</u> choice of F

C is the <u>second</u> choice of D

	+			-
A	B		F	
B		E	F	A
C	D	E		
D	F	C		
E		C		B
F	A	B	D	

Table 3. Stable Roommate Algorithm workflow process

Stable Roommate

A D E

F C

A has more than one choice

F is the <u>second</u> choice of A

D is the <u>last</u> choice of F

C is the <u>second</u> choice of D

E is the <u>last</u> choice of C

\+ -

A	B		F	
B		E	F	A
C	D	E		
D	F	C		
E		C		B
F	A	B	D	

Table 3. Stable Roommate Algorithm workflow process

4

home exchange matching

Stable Roommate

A	D	E
F	C	B

A has more than one choice

F is the <u>second</u> choice of A

D is the <u>last</u> choice of F

C is the <u>second</u> choice of D

E is the <u>last</u> choice of B

B is the <u>second</u> choice of E

\+ -

A	B		F	
B		E	F	A
C	D	E		
D	F	C		
E		C		B
F	A	B	D	

Table 3. Stable Roommate Algorithm workflow process

4

home exchange matching

Stable Roommate

A D E A

F C B

A has more than one choice

F is the second choice of A

D is the last choice of F

C is the second choice of D

E is the last choice of C

B is the second choice of E

A is the last choice of B

\+ ↔ -

A	B		F	
B		E	F	A
C	D	E		
D	F	C		
E		C		B
F	A	B	D	

Table 3. Stable Roommate Algorithm workflow process

Stable Roommate

preference cycle

A D E A

F C B

\+ -

A	B		F	
B		E	F	A
C	D	E		
D	F	C		
E		C		B
F	A	B	D	

Table 3. Stable Roommate Algorithm workflow process

Stable Roommate

A	~~D~~	E	A
~~F~~	C	B	

\+ ⟷ -

A	B		F	
B		E	F	A
C	D	E		
D	~~F~~	C		
E		C		B
F	A	B	~~D~~	

Table 3. Stable Roommate Algorithm workflow process

Stable Roommate

A ~~D~~ E A

~~F~~ C B

\+ ⟵⟶ -

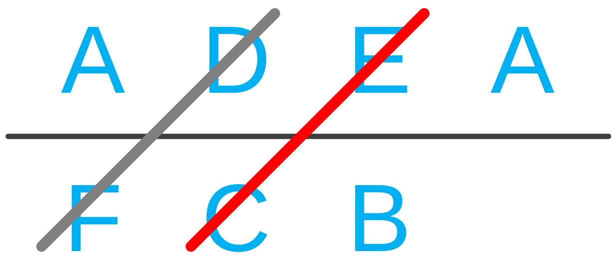
A	B		F	
B		E	F	A
C	D	E		
D		C		
E		C		B
F	A	B		

Table 3. Stable Roommate Algorithm workflow process

4

home exchange matching

Stable Roommate



\+ ↔ -

A	B		F	
B		E	F	A
C	D	~~E~~		
D		C		
E		~~C~~		B
F	A	B		

Table 3. Stable Roommate Algorithm workflow process

Stable Roommate

A ~~D~~ ~~E~~ A

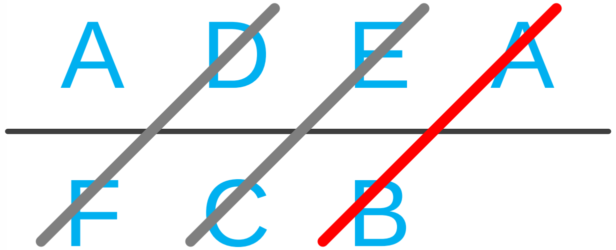
~~F~~ ~~C~~ B

+ ⟷ -

A	B		F	
B		E	F	A
C	D			
D		C		
E				B
F	A	B		

Table 3. Stable Roommate Algorithm workflow process

Stable Roommate



	+			-
A	~~B~~		F	
B		E	F	~~A~~
C	D			
D		C		
E				B
F	A	B		

Table 3. Stable Roommate Algorithm workflow process

Stable Roommate

A ~~D~~ ~~E~~ ~~A~~

~~F~~ ~~C~~ ~~B~~

\+ ⟷ -

A			F	
B		E	F	
C	D			
D		C		
E				B
F	A	B		

Table 3. Stable Roommate Algorithm workflow process

Stable Roommate

B

B has more than one choice

\+ ⟷ -

A			F	
B		E	F	
C	D			
D		C		
E				B
F	A	B		

Table 3. Stable Roommate Algorithm workflow process

Stable Roommate

B

F

B has more than one choice

F is the second choice of B

\+ ⟷ -

A			F	
B		E	F	
C	D			
D		C		
E				B
F	A	B		

Table 3. Stable Roommate Algorithm workflow process

Stable Roommate

B	B
F	

B has more than one choice

F is the <u>second</u> choice of B

B is the <u>last</u> choice of F

\+ -

A			F	
B		E	F	
C	D			
D		C		
E				B
F	A	B		

Table 3. Stable Roommate Algorithm workflow process

4

home exchange matching

Stable Roommate

preference cycle

B	B
F	

\+ ⟷ -

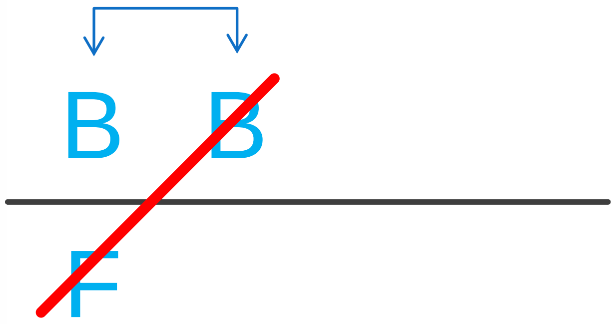
A			F	
B		E	F	
C	D			
D		C		
E				B
F	A	B		

Table 3. Stable Roommate Algorithm workflow process

4

home exchange matching

Stable Roommate



\+ ⟷ -

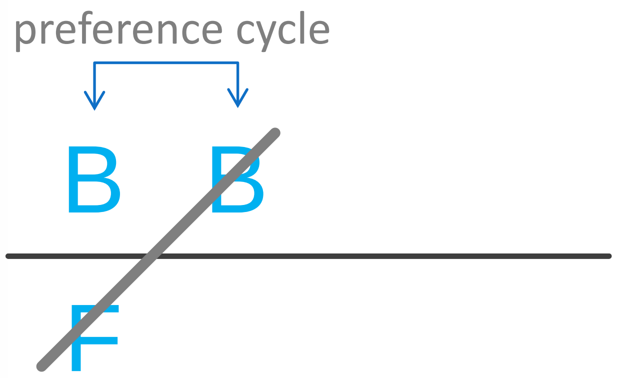
A			F	
B		E	~~F~~	
C	D			
D		C		
E				B
F	A	~~B~~		

Table 3. Stable Roommate Algorithm workflow process

4

home
exchange
matching

Stable Roommate



	+			-
A			F	
B		E		
C	D			
D		C		
E				B
F	A			

Table 3. Stable Roommate Algorithm workflow process

4

home exchange matching

Stable Roommate

A — F

B — E

C — D

\+ ⟷ -

A	F
B	E
C	D
D	C
E	B
F	A

Table 3. Stable Roommate Algorithm workflow process

Stable Roommate

A — F

B — E

C — D

\+ ⟷ -

A	B	D	F	C	E
B	D	E	F	A	C
C	D	E	F	A	B
D	F	C	A	E	B
E	F	C	D	B	A
F	A	B	D	C	E

Table 3. Stable Roommate Algorithm workflow process

4

home exchange matching

```
while there are unmatched agents do

    let a1 be first unmatched agent
    a1 proposes to its first preference a2 who has not rejected it previously

    if a2 has not receieved a proposal before then
        a2 accepts a1

    else
        if a2 prefers a1 over its current match a3 then
            a2 accepts a1
            reject symmetrically (a2, a3)
        else
            reject symmetrically (a1, a2)
        end if
    end if
end while

for all a2 holding proposal from a1 do
    reject symmetrically all (a2, a3) where a2 prefers a1 over a3
end for

for all cycles in (p1...pn+1) and (q1...qn) such that:
    qi is the second preference of pi and pi+1 is the last preference of qi do

    for i = 1 .. n do
        rejects symmetrically (q1, pi+1)
    end for
end for
```

Figure 14. Pseudocode of Stable Roommate Algorithm

housing market matching algorithm

- Gale's the top trading cycle algorithm for the indivisible goods trading such as **housing market** problem [6]
  - agents **own a house**
  - a housing market is an exchange market where agents have the option to trade their house in order to get **a better one**
  - they have preferences over **all houses** **including their own**
  - the agents are allowed to exchange the houses in an **exchange economy**
    - monetary transfers are not available

housing market matching algorithm

- Gale's the top trading cycle algorithm
  - searches for **core cycles** in each iteration
  - allows cycle matching
    - a → b
    - b → c
    - c → a

- In home exchange problem, **mutual exchange mechanism** is needed
  - **cycle matching is not practical**

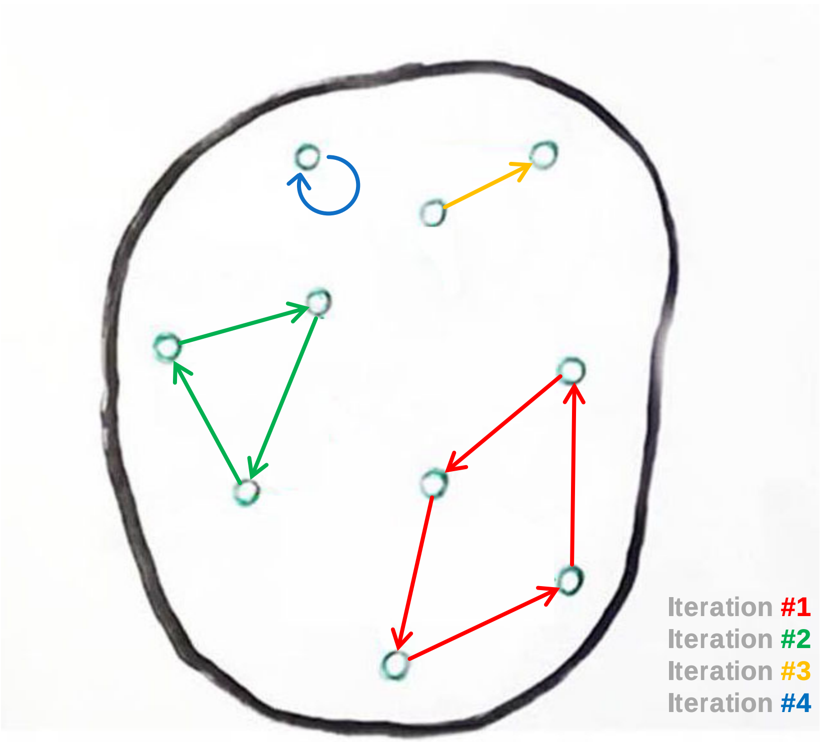
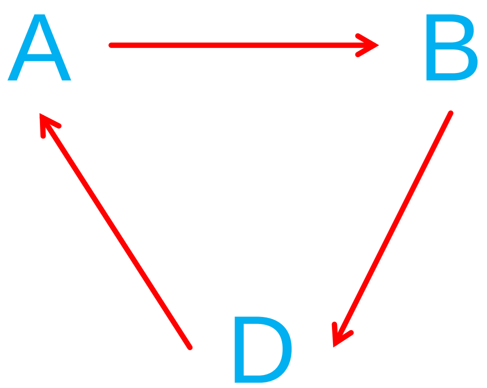


Figure 15. Top Trading Cycles (TTC)

Top Trading Cycles (TTC)



+ ⟷ -

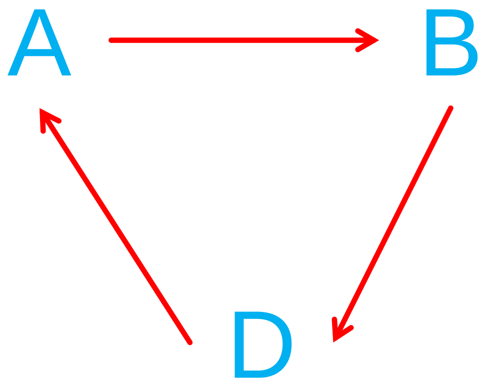
A	B	D	F	C	E
B	D	E	F	A	C
C	D	E	F	A	B
D	A	C	F	E	B
E	F	C	D	B	A
F	A	B	D	C	E

Table 4. Top Trading Cycles (TTC) Algorithm workflow process

4

home
exchange
matching

Top Trading Cycles (TTC)



+ ⟷ -

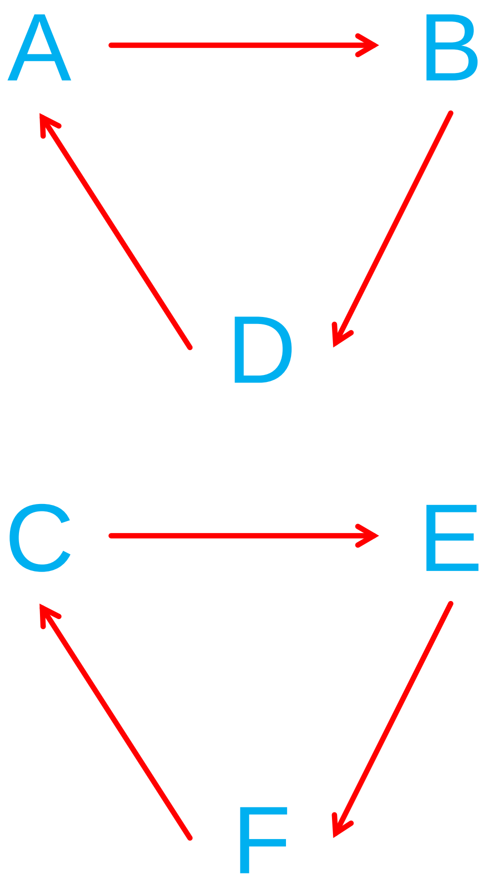
C		E	F		
E	F	C			
F				C	E

Table 4. Top Trading Cycles (TTC) Algorithm workflow process

4

home
exchange
matching

Top Trading Cycles (TTC)



+ -

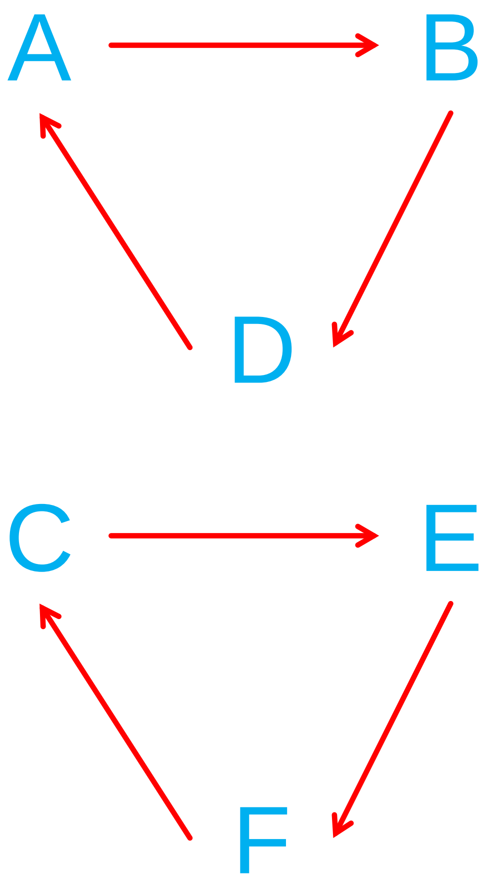
C		E	F		
E	F	C			
F				C	E

Table 4. Top Trading Cycles (TTC) Algorithm workflow process

4

home
exchange
matching

Top Trading Cycles (TTC)



\+ -

Table 4. Top Trading Cycles (TTC) Algorithm workflow process

```
while there are unmatched agents do

    let a1 be first unmatched agent
    empty cycle_pool

    loop
        if cycle_pool contains a1 then
            while a1 is not the first element of cycle_pool do
                remove the first element in cycle_pool
            end while
            break loop
        else
            add a1 to cycle_pool
            let a1 be the first agent in preference list of a1
        end if
    end loop

    foreach agent a1 in cycle_pool
        match a1 with its first_choice asymmetrically
    end for

    foreach agent in matchpool do
        remove matching agents and update preference list
    end for

end while
```

Figure 15. Pseudocode of Top Trading Cycles (TTC) Algorithm

preference-rank matching algorithm

- a problem-specific home exchange matching algorithm has been designed
- **agent-based** approach has been preferred to define and solve home exchange problem
  - each home location is considered as an agent
- each agent **has a preference list** of other agents for home exchange
  - preference list is **sorted by travel times** ascending order
- there is a **threshold level** to take an agent in preference list, which is assigned 10 minutes in this study.
  - a match must reduce travel times at least 10 minutes **for both agents**.

PREFERENCE_RANK

- This value is simply the answer of

  "What is my index number in the preference list of the first agent in my preference list?" question

a1	a2	a3	a4	a5
a2	a1	a1	a3	a1
a5	a4	a2	a2	a3
a3	a3	a5		
		a4		
1	1	3	4	2

Figure 16. Preference Rank value for a4

home exchange matching algorithm

- the PREFERENCE_RANK value must be found for each agent
- the agent with **the smallest** PREFERENCE_RANK value is **paired with its first choice**
  - tie is broken by random selection
- when a match is found,
  - paired agents are removed from matching pool
  - other agents update their preference lists
  - the agents with empty preference list are also removed from matching pool
- the process is repeated until there is no agent in matching pool

Preference-Rank

(4)

	1	2	3	4	5
A	$\boxed{B}$	D	E	C	F
B	D	E	F	A	C
C	D	E	F	A	B
D	F	C	A	E	B
E	F	C	D	B	A
F	A	B	D	C	E

Table 5. Preference-Rank Algorithm workflow process

4

home exchange matching

4

home exchange matching

Preference-Rank

		1	2	3	4	5
(4)	A	B	D	E	C	F
(5)	B	$\boxed{D}$	E	F	A	C
	C	D	E	F	A	B
	D	F	C	A	E	B
	E	F	C	D	B	A
	F	A	B	D	C	E

Table 5. Preference-Rank Algorithm workflow process

Preference-Rank

		1	2	3	4	5
(4)	A	B	D	E	C	F
(5)	B	D	E	F	A	C
(2)	C	$\boxed{D}$	E	F	A	B
	D	F	C	A	E	B
	E	F	C	D	B	A
	F	A	B	D	C	E

Table 5. Preference-Rank Algorithm workflow process

4

home exchange matching

Preference-Rank

		1	2	3	4	5
(4)	A	B	D	E	C	F
(5)	B	D	E	F	A	C
(2)	C	D	E	F	A	B
(3)	D	$\boxed{F}$	C	A	E	B
	E	F	C	D	B	A
	F	A	B	D	C	E

Table 5. Preference-Rank Algorithm workflow process

4

home exchange matching

Preference-Rank

		1	2	3	4	5
(4)	A	B	D	E	C	F
(5)	B	D	E	F	A	C
(2)	C	D	E	F	A	B
(3)	D	F	C	A	E	B
(5)	E	$\boxed{F}$	C	D	B	A
	F	A	B	D	C	E

Table 5. Preference-Rank Algorithm workflow process

Preference-Rank

		1	2	3	4	5
(4)	A	B	D	E	C	F
(5)	B	D	E	F	A	C
(2)	C	D	E	F	A	B
(3)	D	F	C	A	E	B
(5)	E	F	C	D	B	A
(5)	F	$\boxed{A}$	B	D	C	E

Table 5. Preference-Rank Algorithm workflow process

4

home
exchange
matching

4

home
exchange
matching

Preference-Rank

		1	2	3	4	5
(4)	A	B	D	E	C	F
(5)	B	D	E	F	A	C
(2)	C	D	E	F	A	B
(3)	D	F	C	A	E	B
(5)	E	F	C	D	B	A
(5)	F	A	B	D	C	E

Table 5. Preference-Rank Algorithm workflow process

Preference-Rank

		1	2	3	4	5
(4)	A	B	D	E	C	F
(5)	B	D	E	F	A	C
(2)	C	D	E	F	A	B
(3)	D	F	C	A	E	B
(5)	E	F	C	D	B	A
(5)	F	A	B	D	C	E

Table 5. Preference-Rank Algorithm workflow process

4

home exchange matching

4

home exchange matching

Preference-Rank

C — D

	1	2	3	4	5
A	B	D	E	C	F
B	D	E	F	A	C
C	D	E	F	A	B
D	F	C	A	E	B
E	F	C	D	B	A
F	A	B	D	C	E

Table 5. Preference-Rank Algorithm workflow process

4

home
exchange
matching

Preference-Rank

C — D

	1	2	3	4	5
A	B		E		F
B		E	F	A	
E	F			B	A
F	A	B			E

Table 5. Preference-Rank Algorithm workflow process

4

home exchange matching

Preference-Rank

C — D

(3)

	1	2	3	4	5
A	$\boxed{B}$		E		F
B		E	F	A	
E	F			B	A
F	A	B			E

Table 5. Preference-Rank Algorithm workflow process

4

home
exchange
matching

Preference-Rank

C — D

		1	2	3	4	5
(3)	A	B		E		F
(2)	B		E	F	A	
	E	F			B	A
	F	A	B			E

Table 5. Preference-Rank Algorithm workflow process

Preference-Rank

C — D

		1	2	3	4	5
(3)	A	B		E		F
(2)	B		E	F	A	
(3)	E	$\boxed{F}$			B	A
	F	A	B			E

Table 5. Preference-Rank Algorithm workflow process

4

home exchange matching

Preference-Rank

C — D

		1	2	3	4	5
(3)	A	B		E		F
(2)	B		E	F	A	
(3)	E	F			B	A
(3)	F	A	B			E

Table 5. Preference-Rank Algorithm workflow process

4

home
exchange
matching

Preference-Rank

C — D

		1	2	3	4	5
(3)	A	B		E		F
(2)	B		E	F	A	
(3)	E	F			B	A
(3)	F	A	B			E

Table 5. Preference-Rank Algorithm workflow process

Preference-Rank

4

home
exchange
matching

C — D

		1	2	3	4	5
(3)	A	B		E		F
(2)	B		E	F	A	
(3)	E	F			B	A
(3)	F	A	B			E

Table 5. Preference-Rank Algorithm workflow process

4

home
exchange
matching

Preference-Rank

C — D

B — E

	1	2	3	4	5
A	B		E		F
B		E	F	A	
E	F			B	A
F	A	B			E

Table 5. Preference-Rank Algorithm workflow process

4

home exchange matching

Preference-Rank

C — D

B — E

	1	2	3	4	5
A					F
F	A				

Table 5. Preference-Rank Algorithm workflow process

4

home exchange matching

Preference-Rank

C — D

B — E

	1	2	3	4	5
A					F
F	A				

Table 5. Preference-Rank Algorithm workflow process

4

home exchange matching

Preference-Rank

C — D

B — E

A — F

	1	2	3	4	5

Table 5. Preference-Rank Algorithm workflow process

```
foreach a1 in agents do
    foreach a2 in agents do
        if a1 prefers a2 and a2 prefers a1
            add a2 in preference_list of a1
        end if
    end for
end for


while true

    foreach agent in match_pool do
        if preference_list of agent is empty
            remove agent from match_pool
        else
            find the preferred_value
        end if
    end for


    if there are no agents in match_pool
        break


    m1 = agent which has the smallest preferred_value
    m2 = first_choice of m1

    match m1 and m2

    remove m1 and m2 from match_pool

    foreach agent in match_pool do
        remove m1 and m2 in preference_list
    end for

end while
```

Figure 17. Pseudocode of Preference-Rank Algorithm

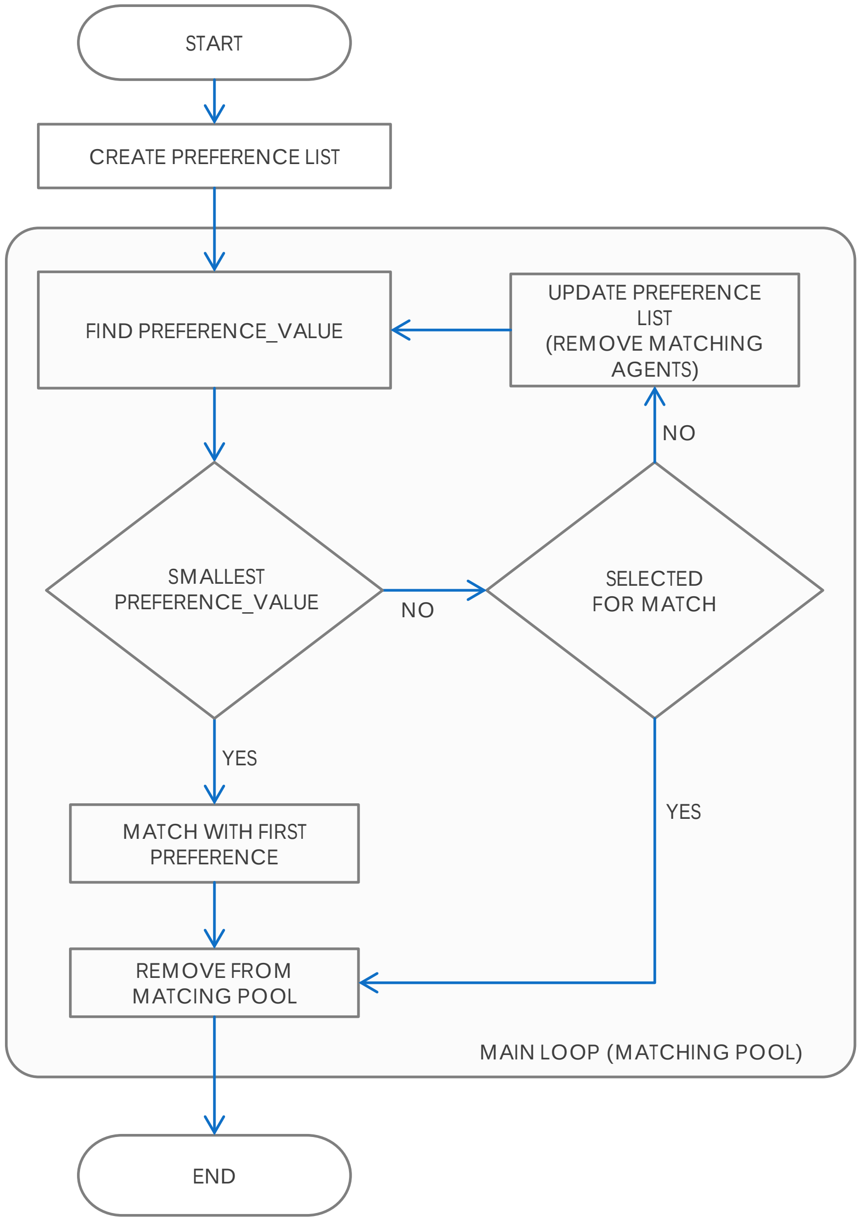


Figure 18. Flowchart of Preference-Rank Algorithm

4

home exchange matching

results

```
 67:[41.007699,28.907034]  3927  2082  1845    01:05:27  00:34:42  00:30:45
102:[41.046367,28.980021]  2347  1628   719    00:39:07  00:27:08  00:11:59

 65:[41.049628,28.902819]  2429  1169  1260    00:40:29  00:19:29  00:21:00
139:[41.131613,29.028815]  2956  2075   881    00:49:16  00:34:35  00:14:41

 58:[41.040409,28.881815]  4342  2818  1524    01:12:22  00:46:58  00:25:24
148:[41.074586,29.037992]  3235  2538   697    00:53:55  00:42:18  00:11:37

 60:[40.998398,28.893576]  2999  1067  1932    00:49:59  00:17:47  00:32:12
104:[41.047918,28.982896]  2124  1413   711    00:35:24  00:23:33  00:11:51

 93:[41.072382,28.952554]  2256   907  1349    00:37:36  00:15:07  00:22:29
 87:[41.015240,28.948254]  2583  1351  1232    00:43:03  00:22:31  00:20:32


31 Matches Have Been Found.
Matching Ratio 41,3

Among Matching Agents (62)
Improvement 54,47
Average Travel Time Before Match  00:53:04
Average Travel Time After Match  00:24:10

In Whole System (150)
Improvement 26,41
Average Travel Time Before Match   00:44:00
Average Travel Time After Match  00:32:22
```

Figure 19. Screenshot of the program output.

4

home
exchange
matching

results

before match

after match

difference

values in time format

agent-67
agent-102
GPS coordinates

```
 67:[41.007699,28.907034]  3927  2082  1845    01:05:27  00:34:42  00:30:45
102:[41.046367,28.980021]  2347  1628   719    00:39:07  00:27:08  00:11:59

 65:[41.049628,28.902819]  2429  1169  1260    00:40:29  00:19:29  00:21:00
139:[41.131613,29.028815]  2956  2075   881    00:49:16  00:34:35  00:14:41

 58:[41.040409,28.881815]  4342  2818  1524    01:12:22  00:46:58  00:25:24
148:[41.074586,29.037992]  3235  2538   697    00:53:55  00:42:18  00:11:37

 60:[40.998398,28.893576]  2999  1067  1932    00:49:59  00:17:47  00:32:12
104:[41.047918,28.982896]  2124  1413   711    00:35:24  00:23:33  00:11:51

 93:[41.072382,28.952554]  2256   907  1349    00:37:36  00:15:07  00:22:29
 87:[41.015240,28.948254]  2583  1351  1232    00:43:03  00:22:31  00:20:32


31 Matches Have Been Found.
Matching Ratio 41,3

Among Matching Agents (62)
Improvement 54,47
Average Travel Time Before Match  00:53:04
Average Travel Time After Match  00:24:10

In Whole System (150)
Improvement 26,41
Average Travel Time Before Match   00:44:00
Average Travel Time After Match  00:32:22
```

Figure 19. Screenshot of the program output.

results

	Matching		Among Matching Agents Commute times			In Whole System Commute times		
Matching Algorithm	Agents	Ratio	Before Match (min)	After Match (min)	Improvements (%)	Before Match (min)	After Match (min)	Improvements (%)
Stable Roommates	63,28	42,19	53:12	27:52	47,65	44:39	33:58	23,88
Preference-Rank	63,29	42,19	53:09	27:50	47,65	44:39	33:58	23,87
Top Trading Cycles	62,99	42	53:14	27:32	48,26	44:39	33:52	24,10

Table 6: Matching algorithms results of 150 home and 25 university locations (average of 1000 successful runs)

4

home
exchange
matching

results

```
for (int k = 0; k < 7; k++)
{
    for (int i = 0; i < 180; i++)
    {
        List<int> generatedValues = new List<int>();

        for (int j = 0; j < 40; j++)
        {
            foreach (var realValue in realValues)
            {
                int t = realValue + random.Next(-300, 300);
                if (t < 300) t = 300;
                generatedValues.Add(t);
            }
        }
    }
}
```

Figure 20. C# programming language code used in virtual data generation process

results

			Among Matching Agents			In Whole System		
	Matching		Commute times			Commute times		
Matching Algorithm	Agents	Ratio	Before Match (min)	After Match (min)	Improvements (%)	Before Match (min)	After Match (min)	Improvements (%)
Stable Roommates	513,38	51,34	50:08	24:20	51,46	43:15	30:00	30,61
Preference-Rank	518,88	51,89	49:57	24:15	51,45	43:15	29:54	30,83
Top Trading Cycles	505,25	50,53	50:18	23:16	53,73	43:15	29:35	31,56

Table 7: Matching algorithms results of 1000 home and 1000 university locations (average of 1000 successful runs)

results

- **30,83%** improvement in whole system
  51,45% improvement among only matching agents are observed

- Average one-way commute time **among matching agents** is reduced from **49 minutes 57 seconds** to **24 minutes 15 seconds**.

- Average one-way commute time **in whole system** is reduced from **43 minutes 15 seconds** to **29 minutes 54 seconds**.

discussion

- All those three algorithms has proved
  more than **50 percent** improvement among matching agents and
  more than **30 percent** improvement in whole system.

discussion

- Home exchange model that is proposed in this study can also be applied on **tenants**.
  - Considering these improvement rates, in the case that the study is extended with tenants, significant improvement rates can be achieved for both matching agents and on whole system.
    - In this way, this model may contribute to reduce commuting times in large populated cities.

- There are some certain **limitations** about home exchange model.
  - in this model, only commute times and physical conditions are considered.
  - there may be custom preferences to choose a location, such as
    - being close to parents, families, friends or
    - school district etc.

Source Codes and Database Files

https://github.com/yusufbuyruk/home-exchange

references

1. https://moovitapp.com/insights/en/Moovit_Insights_Public_Transit_Index_Turkey_Istanbul-1563, 20.10.2018
2. https://developers.google.com/maps/documentation/distance-matrix/intro, 20.10.2018
3. Sönmez, T., & Ünver, M. U. (2011). Matching, allocation, and exchange of discrete resources. In *Handbook of social Economics* (Vol. 1, pp. 781-852). North-Holland.
4. Gale, D., & Shapley, L. S. (1962). College admissions and the stability of marriage. *The American Mathematical Monthly*, *69*(1), 9-15.
5. Irving, R. W. (1985). An efficient algorithm for the “stable roommates” problem. *Journal of Algorithms*, *6*(4), 577-595.
6. Shapley, L., & Scarf, H. (1974). On cores and indivisibility. *Journal of mathematical economics*, *1*(1), 23-37.