ATLAS INTERNATIONAL REFEREED JOURNAL ON SOCIAL SCIENCES
Open Access Refereed E-Journal & Refereed & Indexed
ISSN:2619-936X

January, Vol:5, Issue:16	2019	pp.1-9
Article Arrival Date: 01.12.2018		Published Date: 14.01.2019

SUMO SİMÜLASYON ARACINDA YOL HESAPLAMA ALGORİTMALARININ PERFORMANS ANALİZİ

PERFORMANCE ANALYSIS OF PATH COMPUTING ALGORITHMS IN SUMO SIMULATION TOOL

Gulkaiyr Kalybek KYZY
Institute Of Natural Sciences, Department of Computer and Information Engineering, Faculty of Computer and Information Engineering, Sakarya University, Sakarya, Turkey

Prof.Dr. Cemil ÖZ
Bilgisayar ve Bilişim Bilimleri Fakültesi, Bilgisayar Mühendisliği Bölümü Sakarya, Turkey

Doi Number : http://dx.doi.org/10.31568/atlas.258
Article Type : Review Article

ÖZET

Trafik sorunu büyük şehirlerdeki insanlar için devam eden en önemli sorunlardan birisidir. İşe, okula vb yerlere varış ve çıkış saatlerindeki yoğunluk nedeniyle trafikte çok fazla zaman harcanmaktadır. Son zamanlarda artan trafik sıkışıklığının şiddetli etkisinden dolayı, insanlar trafikte zaman kaybı ve maaliyet artışı yaşamaktadır. Gerçek zamanlı trafik akışları veya geçmiş verilere dayanarak araçların seyahat rotaların optimize etmek için rota hesaplama algoritmaları önerilmiştir. En kısa yolu bulmak için Dijkstra algoritması, A * algoritması, Genetik algoritma, Floyd algoritması ve Karınca kolonisi algoritması gibi birçok algoritma kullanılır. Bu çalışmanın amacı, simülasyon yoluyla araçların hedeflerine ulaşmaları için en iyi yönlendirme algoritmalarının hangileri olduğunu bulmaktır. Çalışmada iki algoritma üzerinde durulacaktır. Bu nedenle, iki algoritma türünün performans analizlerini karşılaştırarak hangisinin az kenarlara (uç) giderek en hızlı ve en az maliyetli şekilde amaca ulaşmasını araştırmaktır. Çalışma görsel SUMO simulasyon aracını kullanarak gerçekleştirilmiştir. Sonuca göre A* algoritmasının arama süresinin Dijkstra'nın algoritmasından daha hızlı olduğunu göstermiştir.

Anahtar Kelimeler: Dijkstra algoritması, A* algoritması, SUMO, Rota hesaplama

ABSTRACT

Traffic is one of the most important issues for people in big cities. We are spending too much time in traffic due to the intensity when we are going and coming from work, school or etc. Due to the serious consequences of the increase in traffic jams people are experiencing as well as the loss of time and a rise in traffic costs. Algorithms for calculating routes are proposed to optimize vehicle routes based on real-time traffic flows or historical data. To find the shortest path used many algorithms such as Dijkstra algorithm, A * algorithm, Genetic algorithm, Floyd algorithm and Ant colony algorithm. This study aims to find out which of the best routing algorithms are tools to achieve their targets through simulation, and the study will focus on two algorithms. Therefore, by comparing the performance analysis of the two types of algorithms, it is necessary to investigate which one reaches the least edges and reach the goal in the least cost way. The study was performed using the visual SUMO simulation tool. Based on the result, the A * algorithm showed that the search time was faster than Dijkstra's algorithm.

Key words: Dijkstra algorithm, A * algorithm, SUMO, Route calculation

1. GİRİŞ

Ulaşımda hedeflenen yere gitme kararı vermeden önce, doğru kararlar alınabilmesi için, teknik teoriler, literatür çalışmaları, mesleki değerlendirme veya trafik simülasyonu gibi iyi bir referanslar gerekmektedir. Insanların yönlendirildiği ve araçların modellenen sürecin önemli bir parçası olduğu sistemleri analiz etmek için simülasyon modellerinin oluşturulduğu

birçok uygulama vardır. Bu alandaki ana görev, geçerli konumdan belirlenen hedefe (amaçtan hedefe) en kısa yolu bulmaktır. Bu problemi çözmek için pek çok algoritma bilinmektedir. Bazı simülasyon sistemleri otomatik araç yönlendirmesi sunar, böylece araç en kısa yolu geçerli konumdan uç uca otomatik olarak alır. Amaç, tüm olası çözümleri kapsamlı bir şekilde araştırmak değil, az hesaplama zamanında iyi çözümler bulmaktır.

2. KAYNAKLAR VE GENEL BAKIŞ

Yol bulma, bir nesneyi daha önceki pozisyonundan nihai pozisyona taşıma süreci olarak tanımlanır. Oyunlar ve Sanal Turlar, Sürücüsüz Araçlar, Robot Hareketleri ve Navigasyonları gibi farklı uygulama alanları Yol Bulma Algoritmalarını kullanmaktadır. Çoğu durumda, amaç mümkün olan en kısa yolu bulmaktır ki bu en uygun, yani en kısa, en ucuz veya en basit olanıdır. Bir kişi tarafından seçilen yolu taklit eden yol, en düşük miktarda yakıt gerektiren yol veya A ve B noktasından C noktasına kadar olan birçok kriterler çoğu yol bulma görevinde sıklıkla karşılaşır. En kısa yolu bulmak navigasyon sistemleri, yapay zeka ve bilgisayar simülasyonları ile biten birçok alandaki en zor konulardan birisidir. Bu alanların kendine özgü algoritmaları olmasına rağmen, başarıyla uygulanan birçok genel amaçlı yol bulma algoritması vardır. En kısa yol algoritmaları şu anda yaygın olarak kullanılmaktadır. İnternet en kısa yol algoritmasının genellikle uygulandığı geniş bir alandır (Shen Wang ve dig.,2013). Programlama dünyasına bakarsak, örneğin Dijkstra ve A * algoritmaları gibi optimal bir yol bulma için kullanılabilecek pek çok algoritma vardır. Bununla birlikte, en kısa yol bulma algoritmalarının performans analizi açısından kıyaslama üzerinde çalışmalar yapılmıştır (Sachithraj T ve dig., Michael Alexander Djojo ve dig., 2013: 25-27). Bu çalışmada, araçların hedefe doğru en kısa yolunu bulurken her algoritmanın karakteristiğini daha iyi tanımak ve optimal rotayı araştırmak için uygun bir algoritmaya ihtiyacımız vardır. Bu nedenle, yapılacak çalışmanın amacı rota hesaplama ve kisa yol bulma algoritmanın performansını karşılaştırmak ve hangisi iyi bir yönlendirme algoritması olduğunu belirleyerek sımülasyonunu yapmaktır.

3. SİMÜLASYON İÇİN KULLANILAN ARAÇLAR VE ALGORİTMALAR

3.1. SUMO

SUMO, her bir araç seviyesinde çalışan mikroskopik bir simülatördür (Center, G. A., 2018). Her bir araç simüle edilir. Araçlar önlerindeki diğer araçların hareketlerini algılayarak, ona göre tepki verirler, trafik ışıkları için dururlar, daha yüksek öncelikli bir karayoluna girmeden önce araçların geçmesini bekler ve kendi bireysel yollarını takip ederler. SUMO gerçek hayatla eşdeğer ölçekte simulasyon haritalarını işleyebilir, ancak bu çalışmada daha küçük ölçekle oluşturulan senaryolar kullanılmıştır. İlk olarak, SUMO'ya kesişim, yol, trafik kontrol lambası, rota ve araçların bir listesi verilir. Birden fazla araç türü simüle edilebilir, ancak bu çalışmada simülasyonlar için temel bir araç seçilmiştir. Simülasyon daha sonra kendi başına ilerleyebilir. Simülasyon işlemi üzerinde daha fazla kontrol elde etmek için simülasyon sırasında SUMO'nun TraCI protokolünü kullanarak etkileşim kuruladabilir (D. Krajzewicz ve dig., 2005, Michael Behrisch ve dig., 2011, Vi Tran Ngoc Nha ce dig.,2012). SUMO sadece bir trafik simülatörü değil, kullanıcının bir yol ağını oluşturmasına içe aktarmasına ve ilgili trafik talebini tanımlamasına olanak veren bir uygulama paketidir. Bir ağı OpenStreetMap'den veya VISUM, MATsim veya VISSIM gibi diğer trafik simülatörlerinden almak için "*netconvert*" kullanır. Bir yol ağı ithal edildiğinde ve uygun formata dönüştürüldüğünde, her araç için trafik talebi ve güzergahlar oluşturulmalıdır. Çalışmada rotaları hesaplamak için "*duarouter*" kullanılmıştır. Ek olarak, trafik kazaları, inşaat ve diğer yavaşlamaları simüle etmek için araçların ve yol bölümlerinin azami hızları değiştirilebilir. Bu çalışmada yapılan simülasyonlar için SUMO 0.32.0 sürümü kullanılmıştır.

3.2. Dijkstra Algoritması

Dijkstra'nın algoritması, en kısa yol ağacı üreten, tüm kenarların ağırlıklarının negatif olmayan bir grafik için tek kaynaklı en kısa yol problemini çözen grafik arama algoritmasıdır. Dijkstra algoritması, bir düğümden diğer tüm düğümlere en kısa yolları hesaplar. Bir başka ifadeyle belirli bir başlangıç noktasına göre en kısa yolu belirleyen bir algoritmadır. Bu algoritma, ağırlıklı ve yönlü graflar için geliştirilmiş olup kenarların ağırlık değeri sıfır ya da sıfırdan büyük bir değer olmalıdır (E.W. Dijkstra.1979, 1:269-271).

Pseudo -code, Dijkstra'nın algoritması için şöyledir.

initialize the cost of each node to infinity

initialize the cost of the source to 0

while there are unknown nodes left in the graph

select an unknown node m with the lowest cost

mark *m* as known

for each node *n* adjacent to *m*

if *m*'s cost + cost of (*m*, *n*) < *n*'s cost

n's cost = *m*'s cost + cost of (*m*, *n*)

n's prev path node = *m*

3.3. Algoritma A * (A Yıldız)

A * (" A yıldız" olarak telaffuz edilir), yol bulma ve grafik geçişinde yaygın olarak kullanılan bir bilgisayar algoritmasıdır. Dijkstra algoritmasının bir uzantısıdır. A * algoritması yol boyunca alternatif yol bölümlerinin bir öncelik sırasını koruyarak, bilinen en düşük yolun bir yolunu izler. Herhangi bir noktada, geçiş yapılan yolun bir segmenti, karşılaşılan başka bir yol segmentinden daha yüksek bir maliyete sahipse, daha yüksek maliyetli yol segmentini terk eder ve bunun yerine daha düşük maliyetli yol segmentine geçer. Bu süreç hedefe ulaşılana kadar devam eder. A *, sezgisel yöntemlerle daha iyi performans (zamana bağlı olarak) sağlar (T. H. Cormen,2013). En kısa yolu hesaplamak için sezgisel yol bulgusunu kullanırız. Hesaplama bir denge durumuna dönüşene kadar iterativ olarak yapılir. En kısa yolu hesaplamak için denklem aşağıdaki gibidir:

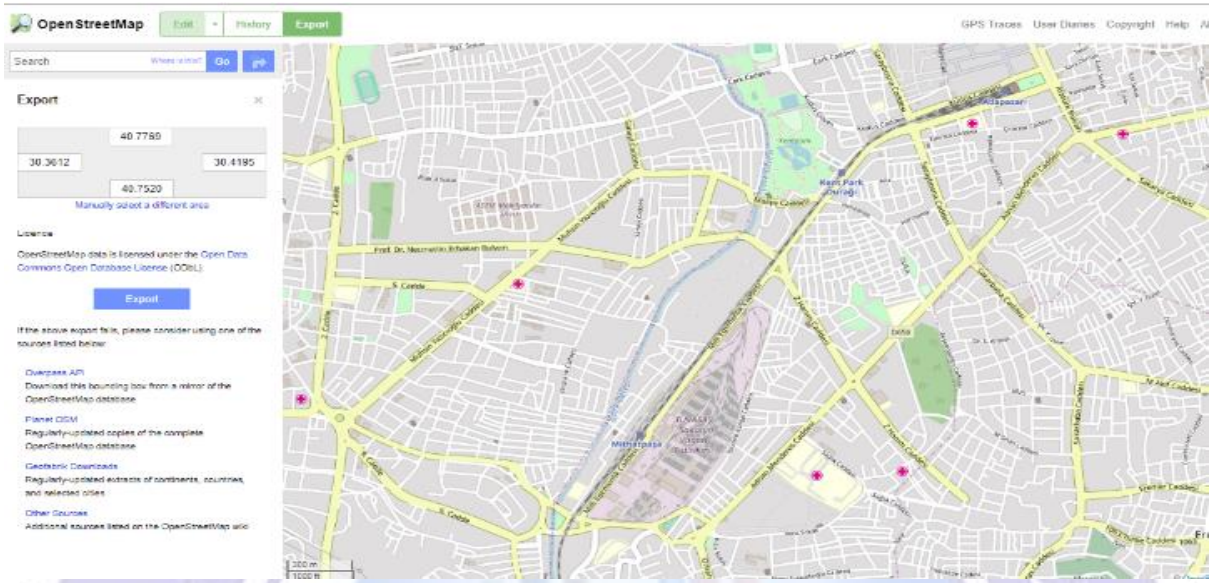
f (n) = g (n) + h (n)

Burda g (n), toplam mesafe ve h (n), hedef düğümün sezgisel mesafesidir, n ise başlangıç noktasıdır.

4. SİMÜLASYON ADIMLARI

Deneyler yürütülmek üzere en yaygın olarak kullanılan açık kaynaklı Kentsel Hareketlilik Simülasyonu (SUMO) 0.32.0(Michael Alexander Djojo,2013) kullanılarak yapılmiştir. SUMO varsayılan olarak Dijkstra algoritmasında çalıştığı için ona yol hesaplama A* algotritması eklenmiştir. Tüm algoritmalar Python'da uygulanmakta ve daha sonra yapılan simülasyonu kullanarak Dijkstra ve A* algoritması uygulanmıştır. Simülasyonu gerçekleştirmek için senaryolar oluşturulmuştur. OpensteetMap'tan açık kaynaklı osm uzantılı Adapazarı haritası indirilmiştir. SUMO'da gerçek durumu andıran geometri ve sokak topolojisine dayanan simülasyonu yapmak için, harita openstreetmap.org'dan bir tanesi içe aktarılarak elde edilebilir (OpenStreetMap). Bu işlem openstreetmap sitesine erişerek ve ardından gerekli yeri seçerek ve sonra dışa aktar'ı tıklayarak yapılabilir (Şekil1). Yol ağının yapısal bağlantısına ek olarak, her yol segmenti şerit sayısı, geometrik şekiller, sokak isimleri,

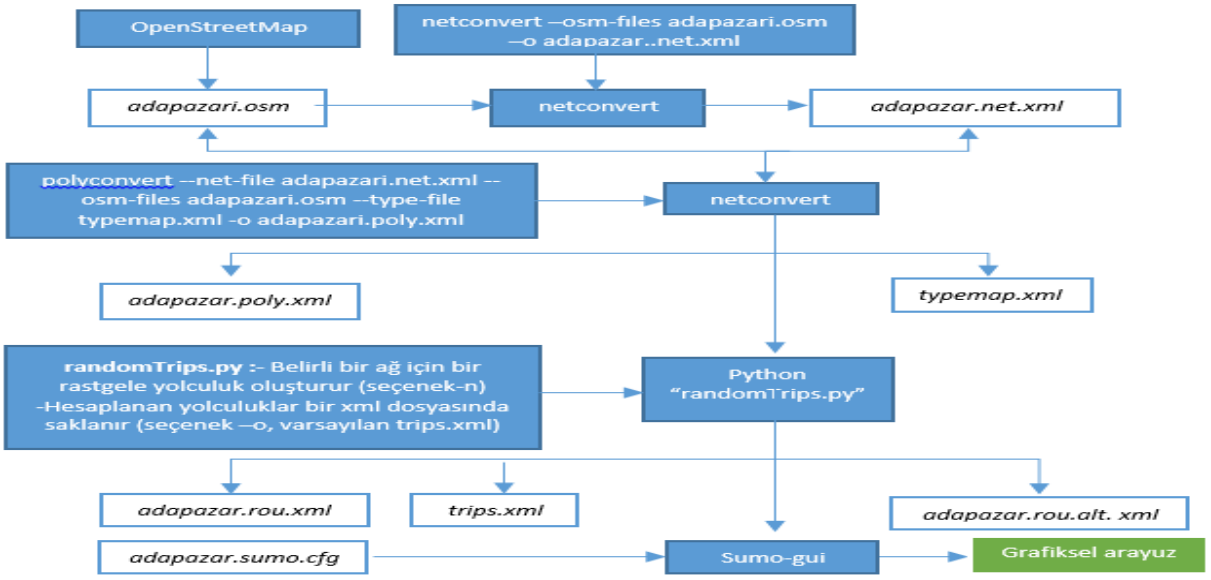
sokak tipi, yolun uzunluğu, dönüş şeritleri ve her şeridin hız sınırı gibi özellikleri de içerir. Seçtiğimiz bölge için osm uzantılı dosya elde ettikten sonra XML dosyasına yani başka bir formata dönüştürmek için SUMO Netconvert kullanılmıştır. Sonra, bu dosya SUMO'da çalıştırılmak üzere python programlaması kullanılarak ağ dosyasına ve poligona dönüştürülmüştür (Behrisch, M ve dig., 2014).



Şekil 1. OpenStreetMap

Şekil 1'de gösterilen haritayı modellemek için gerekli olan harita koordinat bilgileri OpenStreetMap ile elde edildikten sonra Şekil 2 deki adımlar yapılmıştır. Öncelikle Şekil 1'deki osm dosyası XML formatına dönüştürülmesi gerekir (Şekil 2).

- ✓ adapazari.net.xml
- ✓ adapazari.osm
- ✓ adapazari.poly.xml
- ✓ adapazari.rou.alt.xml
- ✓ adapazari.rou.xml
- ✓ "netconvert" kullanarak adapazari.sumo.cfg Uygulama, böylece SUMO tarafından okunabilir.



Şekil 2. Openstreetmap verilerini SUMO ağ dosyası oluşturma adımları

Sonra adapazarı.net.xml (Şekil 3) ve trips.xml (Şekil 4) dosyaları adapazari.sumo.cfg konfigürasyon dosyası simülatör üzerinde çalıştırılır.

Şekil 3. Adapazari.net.xml

Son olarak, bu harita bilgileri, daha sonra uygulanacak her bir trafik senaryosuna göre randomTrips.py kodu üzerinde değişiklik yapılarak araçların hareket bilgilerini içeren "trips.xml"(Şekil 4) dosyaları oluşturulmuştur.

Open — trips.xml — ~/Downloads/scenarios — Save

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- generated on 2018-12-13 08:59:02.717955 by $Id$
  options: -n adapazari.net.xml -r adapazari.rou.xml -o trips.xml -e
1000 -l
-->
<routes xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://sumo.dlr.de/xsd/routes_file.xsd">
    <trip id="0" depart="0.00" from="-637110636#3" to="-215383114#1"/>
    <trip id="1" depart="1.00" from="140007539#1" to="140126327#0"/>
    <trip id="2" depart="2.00" from="215255392#0" to="-215246147#0"/>
    <trip id="3" depart="3.00" from="491292379#1" to="-215378042#0"/>
    <trip id="4" depart="4.00" from="-215255719#1" to="-548201254#3"/>
    <trip id="5" depart="5.00" from="-215368887#2" to="216717239#4"/>
    <trip id="6" depart="6.00" from="1947440 74#0" to="-2121482 46#0"/>
    <trip id="7" depart="7.00" from="637680656#0" to="324856 48#0"/>
    <trip id="8" depart="8.00" from="215368867#0" to="215255747#0"/>
    <trip id="9" depart="9.00" from="-212147838#4" to="-215400824#0"/>
    <trip id="10" depart="10.00" from="215246113#5" to="215255752#6"/>
    <trip id="11" depart="11.00" from="-215377989#0" to="215227838"/>
    <trip id="12" depart="12.00" from="337043748#0" to="491330739#2"/>
    <trip id="13" depart="13.00" from="-212149322" to="212149514#1"/>
    <trip id="14" depart="14.00" from="1399623 54#1" to="-548201254#3"/>
    <trip id="15" depart="15.00" from="-212148853" to="212148738"/>
    <trip id="16" depart="16.00" from="-215368847#1" to="551834699"/>
    <trip id="17" depart="17.00" from="-637677255#1" to="-215233506#0"/>
    <trip id="18" depart="18.00" from="89795741#0" to="-215372352"/>
    <trip id="19" depart="19.00" from="215255466" to="212149457#4"/>
    <trip id="20" depart="20.00" from="215227877#3" to="215227797"/>
    <trip id="21" depart="21.00" from="-215255418" to="212148257"/>
```

XML — Tab Width: 8 — Ln 6, Col 47 — INS

Şekil 4. Araçların amaçtan hedefe doğru bilgileri, trip.xml

4.1. Grafik Arayüzü

SUMO' da trafik simülasyonu iki farklı şekilde yürütülebilir:

- ✓ Komut satırını doğrudan kullanarak:

sumo-gui haritası adapazari.sumo.cfg

- ✓ SUMO-GUI uygulaması ile

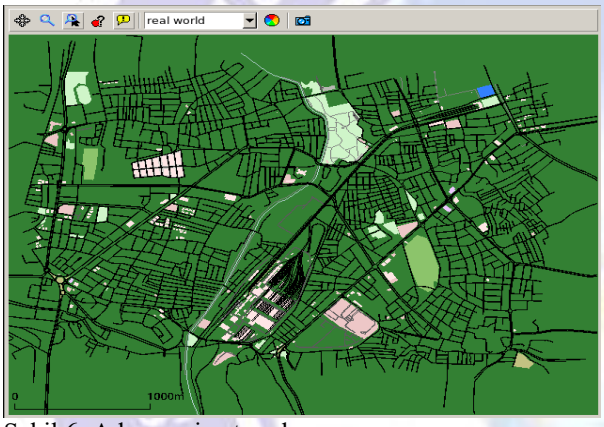
Grafiksel arayüzün ekranı temiz ve çeşitli tiplerde görünür: çok basit, standart veya daha gerçekçi bir görünüm altındadır. Simülasyonun yürütme hızını ayarlama olasılığı her zaman

mevcuttur, bu bir program adımına karşılık gelen milisaniye sayısını tanımlayarak ayarlanabilir. Uygulama sırasında, her araç hareketi ve trafik ilerlemesi gözlemlenebilir (Şekil 5).

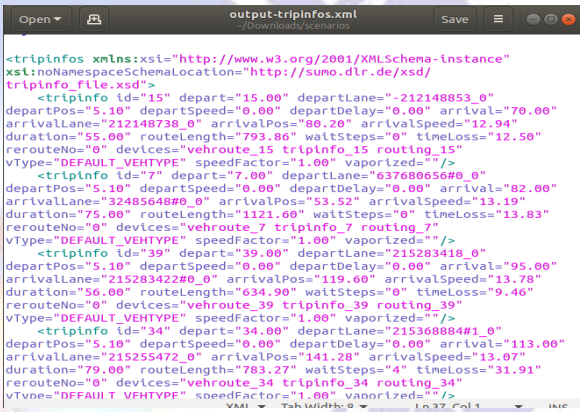
Şekil 5. Görsel Simülasyon yakınlaştırma görünümü

4.2. Simulasyon Sonuçları

SUMO, her simülasyon çalışması için çeşitli çıktılar (*.out.xml) üretmeye izin verir, görselleştirme SUMO-GUI (Şekil 6) kullanılarak yapılır. Simülasyonun çıkışı ortalama bekleme süresi, seyahat bilgileri kullanılarak yapılmıştır. Bu dosya tipi XML olarak oluşturur ve Şekil 7 deki gibi < output-tripinfos.xml > olarak adlandırılır.



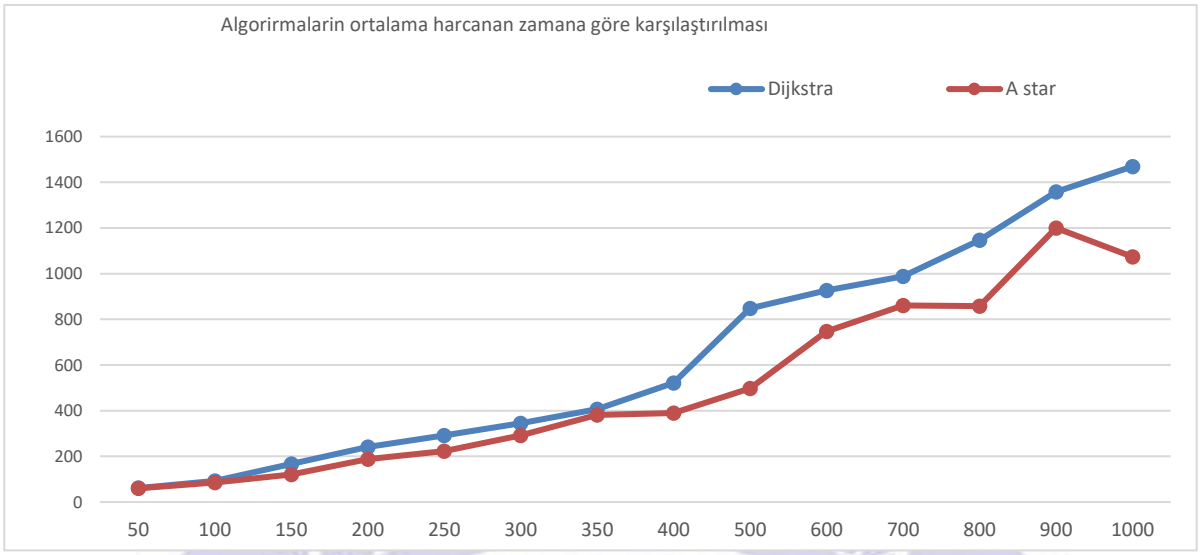
Şekil 6. Adapazari.net.xml



Şekil 7. SUMO progrma çıktısı

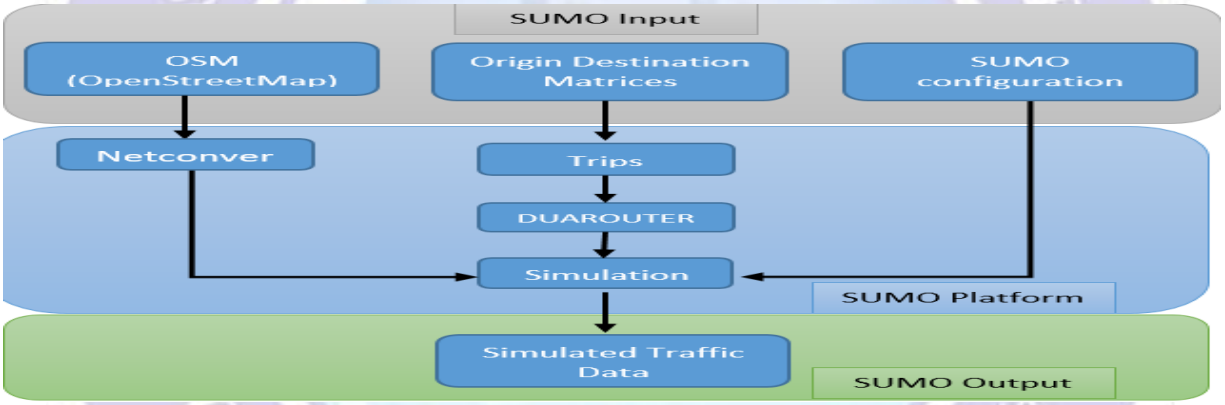
Tablo 1. Harcanan zamana göre algoritmaların sonucşarı

Araç sayıları	Dıjkstra		Astar	
	Ortalama harcanan zaman	Ortalama Kenarlar	Ortalama harcanan zaman	Ortalama Kenarlar
50	61	2650.68	50	1723
100	131	3041.82	104	2044.73
150	217	2880.27	148	1875.60
200	241	2986.38	188	2030.92
250	292	3061.58	223	2128.19
300	345	3134.95	291	1985,23
350	407	3035.66	381	2077,59
400	521	3005,18	390	1991.03
500	848	3040,29	498	2079,47
600	926	3157,06	747	2066,25
700	955	3096,05	855	2077,21
800	1146	3173.14	989	2143,84
900	1358	3001,98	1200	2013,6
1000	1469	3187,27	1361	2175.88



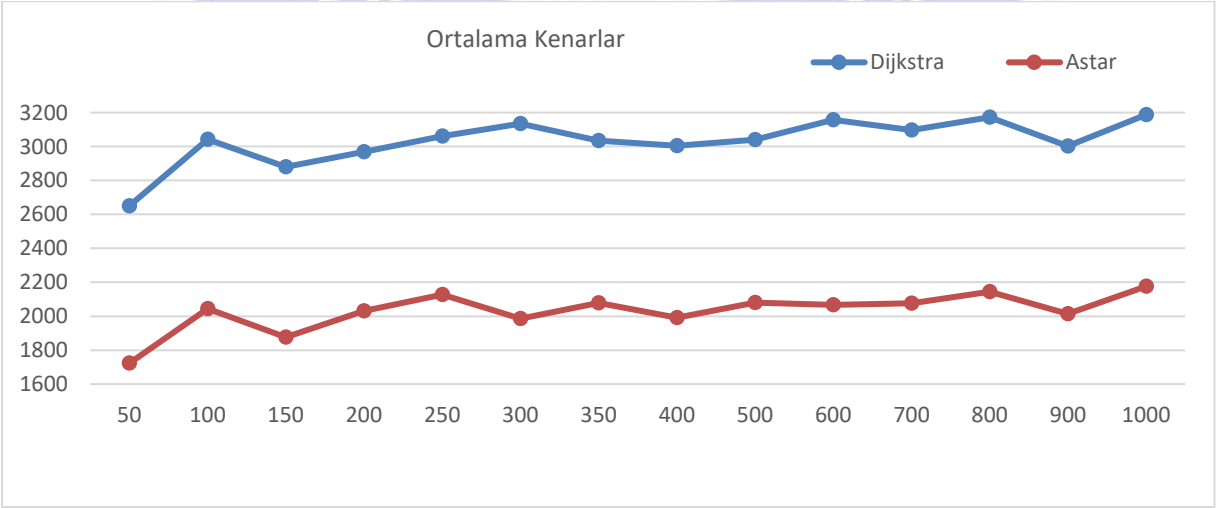
Şekil 8. Harcanan zamana göre algoritmaların karşılaştırılması

Yukarıdaki diyagram (Şekiş 8), toplam harcanan süreleri göstermektedir. Bu sonuçlar, algoritmanın çıktısını temsil etmektedir. Simülasyonu çalıştırdığımızda araçların sayısına göre seyahat sürelerinin değiştiği gözlemlenmektedir. Bu karşılaştırmada giriş parametreleri olarak haritay, araçların sayıları alınmıştır (Şekil9).



Şekil 9. Simülasyon parametreleri

Simülasyonda kullandığımız bölge (harita)toplam 6985 kenardan oluşmaktadır. Simülasyona algoritmalar uygulandığında en fazla 3187 kenara kadar gidilmektedir. Bu araçların sayısına ve verdiğimiz süreye bağlıdır.(Şekil 10)

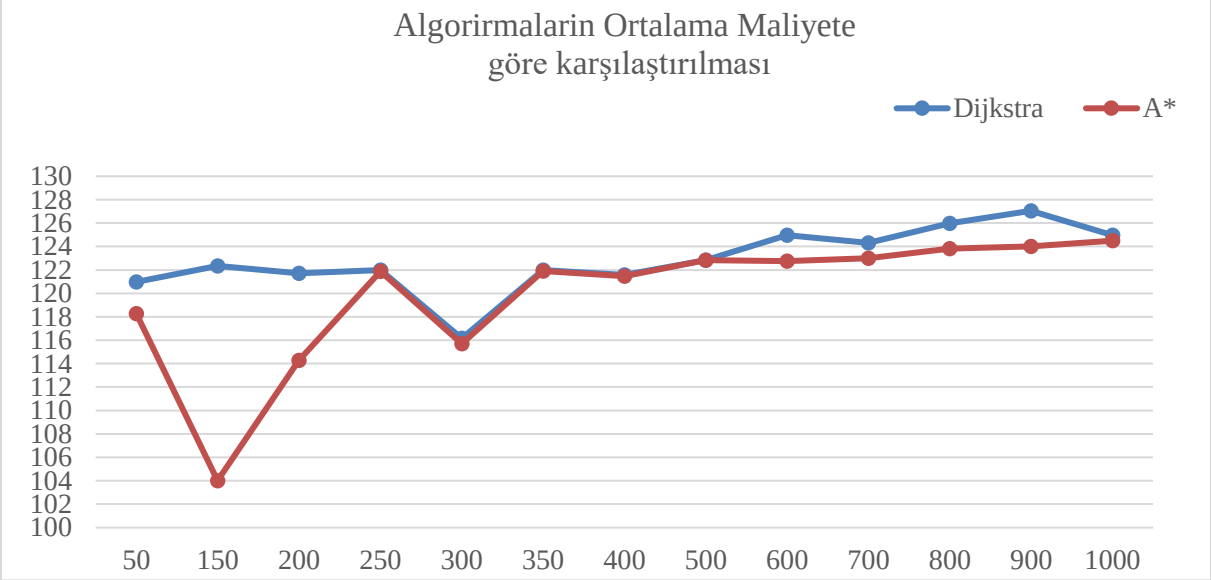


Şekil 10. Ortalama kenarlara göre algoritmaların karşılaştırılması

Simüle edilen rotanın tolpam uzunluğu 499 km dir. En son bilinen maliyetlerle ağdaki araçları yönlendirmek için DUAROUTER'ı çağrılmıştır. Böylece maliyet hesaplamak için SUMO duarouteri kullanıyor ve XML dosyasından aldığımız maliyet sonuçları aracın seyahat süresine dayali hesaplanmıştır. Aşağıdaki tabloda gösterildiği gibi A yıldız algoritması kullanıma uygundur çunki az maliyetle çalışmaktadır.

Tablo 2. Ortalama Maliyet

Ortalama Maliyet		
Araçların sayısı	Dijkstra	A*
50	120.99	118.28
150	122.35	104.01
200	121.72	114.28
250	121.98	121.87
300	116.16	115.69
350	121.98	121.92
400	121.57	121.48
500	122.85	122.85
600	124.98	122.75
700	124.32	123.01
800	125.99	123.81
900	127.05	124.1
1000	124.98	124.50



Şekil 11. Ortalama maliyete göre algoritmaların karşılaştırılması

5. SONUÇ

Bu çalışmada araçların yönlendirme algoritmalarının bir performans değerlendirilerek elde edilen sonuçların karşılaştırılması yapılmıştır. Simülasyon toplam 1000 araç üzerinde yapılmıştır. Yönlendirme algoritmaları için yapılan değerlendirme çalışması, gerçek ulaşım ağına ve son derece gerçekçi trafik yüküne dayalı olarak gerçekleştirildi. Trafik simülasyonu SUMO üzerinde simüle edilmiştir. Araç hızı, kenar sayısı, araç sayısı, yolun uzunluğu, verilen süre gibi güncel durumlarla senkronize etmek için dikkate alınması gereken bazı parametreler içerilmiştir. Kısa yol bulma algoritmalarının içinden iki algoritma seçilerek SUMO ve OpenStreetMap (OSM) kullanarak daha gerçekçi bir durumda şehir trafiği simüle edilmiştir.

Yapılan simülasyonun sonuçlarına dayanarak, şu sonuçlara varılabilir:

- ✓ Dijkstra algoritması ve A* algoritması oldukça hızlı bir sürede en iyi sonuçları sağlar.

- ✓ Dijkstra algoritmasının ve A* algoritmasının harcanan zamanlarını karşılaştırdığımızda A* algoritması en iyi sonuçları sağlamıştır.
- ✓ Değişken tıkanıklık seviyesi ve rastgele trafikteki araçların sayısı değiştirilerek dinamik yol ağlarında Dijkstra ve A* algoritmasının verimliliği simüle edilmiştir.

Bu çalışmada performans değerlendirmesi ve yöntemlerin güvenilirliği önerilmiştir. Performans değerlendirmesi için kıyaslama, hangi algoritmanın yanıt süresi açısından en etkili olduğunu, karşılaştırılan algoritmalar arama verimliliğini etkili bir şekilde artırabildiğini ve en kısa yolu ararken arama düğüm sayısını azaltabildiğini göstermektedir.

Sonuç olarak sonuçlarda gösterildiği gibi en kısa yol bulmak için A * algoritmasın kullanırsak verimli olacaktır.

KAYNAKLAR

Shen Wang, Soufiene Djahel, Jennifer McManis, Cormac McKenna, Liam Murphy and Lero RINCE. (2013)"Comprehensive performance analysis and comparison of vehicles routing algorithms in smart cities", *Global Information Infrastructure Symposium*

Sachithraj T , Piruthiviraj.P , Preeta Sharan, "Comparison Of Dijkstra Algorithm With Ant Colony Optimization Algorithm Using Random Topology In All-Optical Network Using Rwa", *IJRET: International Journal of Research in Engineering and Technology* eISSN: 2319-1163 | pISSN: 2321-7308

Michael Alexander Djojo , Kanisius Karyono, (2013). "Computational load analysis of Dijkstra, A*, and Floyd-Warshall algorithms in mesh network", *2013 International Conference on Robotics, Biomimetics*, Intelligent Computational Systems (ROBIONETICS) Yogyakarta, Indonesia, November 25-27

Center, G. A.(Erişim tarihi: Aralik, 2018) SUMO simulation of urban Mobility. http://sumo.sourceforge.net/ ,

D. Krajzewicz, M. Hartinger, G. Hertkorn, P. Mieth, C. Rıssel, J. Zimmer and P. Wagner, (2005). "Using the Road Traffic Simulation "SUMO" Educational Purposes" *Institute for Transportation Research – German Aerospace Centre*, Rutherfordstr. 2, 12489 Berlin – Germany

Michael Behrisch, Laura Bieker, Jakob Erdmann, Daniel Krajzewicz (2011)"SUMO – Simulation of Urban Mobility", *Institute of Transportation Systems, German Aerospace Center* , Rutherfordstr. 2, 12489 Berlin, Germany

E.W. Dijkstra. (1979)"A note on two problems in connexion with graphs", Numerische Matematik, 1:269-271

T. H. Cormen, (2013) "Algorithms Unlocked," *MIT Press*, Cambridge

OpenStreetMap, (Erişim tarihi: Aralık, 2018) http://www.openstreetmap.org/

Behrisch, M., Krajzewicz, D. and Weber, (2014)M. "Simulation of Urban Mobility"

Vi Tran Ngoc Nha, Soufiene Djahel and John Murphy Lero, (2012)"A Comparative Study of Vehicles' Routing Algorithms for Route Planning in Smart Cities" , UCD *School of Computer Science and Informatics*, Ireland