Gazi Üniv. Müh. Mim. Fak. Der.
Cilt 26, No 4, 915-928, 2011

J. Fac. Eng. Arch. Gazi Univ.
Vol 26, No 4, 915-928, 2011

ZAMANA BAĞLI DİNAMİK EN KISA YOL PROBLEMİ İÇİN GENETİK ALGORİTMA TABANLI YENİ BİR ALGORİTMA

Murat DENER*, M. Ali AKCAYOL, Sinan TOKLU** ve Ömer Faruk BAY***
* Elektronik ve Bilgisayar Eğitimi, Teknoloji Fakültesi, Gazi Üniversitesi, Ankara
** Bilgisayar Mühendisliği, Mühendislik Fakültesi, Gazi Üniversitesi, Ankara
muratdener@gazi.edu.tr, akcayol@gazi.edu.tr, s.toklu@gazi.edu.tr, omerbay@gazi.edu.tr

(Geliş/Received: 04.05.2010; Kabul/Accepted: 10.08.2011)

ÖZET

En kısa yol problemi için çok sayıda araştırma yapılmakta ve ortaya konulan çözümler başta bilgisayar mühendisliği ve endüstri mühendisliği olmak üzere çok farklı alanlarda uygulanmaktadır. Bu makalede yol maliyetleri zamana bağlı dinamik olarak değişen en kısa yol probleminin çözümü için genetik algoritma kullanılarak yeni bir algoritma geliştirilmiştir. Önerilen algoritma ile literatürde yer alan algoritmaların karşılaştırılması için örnek bir uygulama geliştirilmiştir. Benzetim sonuçları geliştirilen algoritmanın daha başarılı olduğunu göstermiştir.

Anahtar Kelimeler: Genetik algoritma, En kısa yol problemi, Ağlar

GENETIC ALGORITHM BASED A NEW ALGORITHM FOR TIME DYNAMIC SHORTEST PATH PROBLEM

ABSTRACT

Many studies have been done for the shortest path problem and the results of these studies have been applied to different areas especially computer engineering and industrial engineering. In this study, a new algorithm has been developed using genetic algorithm for shortest path problem which have dynamic path cost depending on time. An example case study have been developed to compare proposed algorithm and the others. The simulation results show that the proposed algorithm is more successful than the others.

Keywords: Genetic algorithm, Shortest path problem, Networks

1. GİRİŞ (INTRODUCTION)

Günümüzde, teknoloji hızla gelişmektedir. Bu teknolojik gelişimin sonucu olarak günlük hayatta yapılan bir çok iş daha kolay ve kısa sürede gerçekleştirilmektedir. Trafik problemi büyük şehirlerde insanlar için en önemli sorunlardan birisidir. İşe geliş ve çıkış saatlerinde meydana gelen aşırı yoğunluk nedeniyle trafikte çok zaman harcanmaktadır. Bunun sonucu olarak, hem zaman kaybı olmakta, hem maliyet artışı olmakta, hem de insanlar trafikte yorulmakta ve yıpranmaktadırlar.

En kısa yol problemi çok eski bir problemdir ve bu alanda çok sayıda çalışma yapılmıştır. İnsanlar bir noktadan bir başka noktaya daha hızlı ve kolay nasıl gidebileceklerini araştırmışlardır.

En kısa yolu bulma probleminin amacı önceden belirlenen başlangıç noktasıyla, hedef nokta arasındaki en az yol maliyetini belirlemektir. Minimum yol maliyetini bulmak bu problemin ana amacıdır. En kısa yolu bulma problemi ağ optimizasyonlarında yoğun olarak kullanılmaktadır [1-3].

Genetik algoritmalar, doğada gözlemlenen evrimsel sürece benzer bir şekilde çalışan arama ve eniyileme yöntemidir [4]. Karmaşık çok boyutlu arama uzayında en iyinin hayatta kalması ilkesine göre en iyi çözümü aramaktadır [5]. Genetik algoritmalar problemlerin çözümü için evrimsel süreci taklit ederler. Diğer eniyileme yöntemlerinde olduğu gibi çözüm için tek bir yapının geliştirilmesi yerine, böyle yapılardan meydana gelen bir küme oluştururlar. Kümedeki

bireyler evrimsel süreç içinde genetik algoritma işlemcileri tarafından belirlenirler [6].

Bu çalışmada en kısa yol probleminin çözümü için genetik algoritma tabanlı yeni bir algoritma önerilmiştir. Genetik algoritmalar, diğer popülasyon tabanlı algoritmalara göre daha fazla operatör kullanır ve karmaşık çok boyutlu arama uzayında en iyinin bulunmasında diğer algoritmalara göre daha etkilidir [5]. Diğer yöntemlerden farklı olarak çözüm için çok sayıda birey üzerinde her iterasyonda birçok operatörü kullanır [4,5].

Bu çalışmada genetik algoritma ile Ankara, Kızılay bölgesi için iki nokta arasında en kısa süreli yolun bulunması gerçekleştirilmektedir. Belirli sürelerde yollara ait yoğunluklar bilindiğinde, trafikte seyreden araçlar kısa sürede istediği yerlere ulaşabilecektir. Trafikte karşılaşılan en istenmeyen durum, bazı yollarda yoğunluktan dolayı araçlar ilerleyemezken, bazı yollarda ise yoğunluğun yok denecek kadar az olmasıdır. Bu çalışma için Ankara'da trafik yoğunluğunun en fazla olduğu Kızılay bölgesi model olarak alınmıştır. Gerçekleştirilen sistemde, sürücüler gidecekleri yöndeki trafik yoğunluğu hakkında bilgi alabilecek ve gerekli hallerde alternatif güzergahları tercih edebileceklerdir. Alternatif yolların tercih edilmesiyle, trafik yoğunluğu dağıtılmış olacak ve trafik sıkışıklığı azalmış olacaktır.

Önerilen sistemin amacı, trafik akışının daha hızlı ve güvenli olmasına katkı sağlayarak trafik yoğunluğunu azaltmak, sürücülere zaman kazandırmak ve yakıt tasarrufu sağlamaktır.

Makalede, trafikte en kısa süreli yolu bulma problemi hakkında bilgiler ve problemin genetik algoritma ile çözümü Bölüm 2'de verilmiştir. Bölüm 3'te önerilen sistemin tasarımı detaylı bir şekilde anlatılmıştır. Bölüm 4'te benzetim sonuçları sunulmuş ve Bölüm 5'de ise yapılan çalışmanın genel sonuçları verilmiştir.

1.1. İlgili Çalışmalar (Related Works)

En kısa yol problemin modellenmesi ve çözümü için çok sayıda araştırma yapılmıştır [9,10]. En kısa yolu bulmak için Bellman [11], Dijkstra [12], Dreyfus [13] tarafından etkili algoritmalar geliştirilmiştir. Wook ve Ramakrishna [14], bu probleme genetik algoritma kullanarak çözüm önermiştir. Wook'a göre popülasyon sayısının fazla olması iyidir fakat; bununla birlikte arama zamanı artmaktadır. Popülasyon sayısı az olunca ise etkili bir çözüm bulunamamaktadır. Bundan dolayı Wook'un algoritmasında başlangıç popülasyonunu oluşturmak için rastgele başlama yöntemi kullanılmıştır. Çaprazlama yaparken kromozomların belirlenmesinde turnuva seçimi yöntemi kullanılmıştır. Turnuva seçiminde rastgele seçilen iki bireyden uygunluk derecesi yüksek olan bir sonraki popülasyona aktarılmaktadır. Bu işlem popülasyondaki kromozom sayısı kadar tekrarlanmaktadır. Fakat; seçim işleminde elitizm uygulanmadığından en iyi bireyin bir sonraki jenerasyonda olmama ihtimali de bulunmaktadır. Faklı iki kromozom arasında mutasyon işlemi de yapılmıştır. Fakat kromozomların yapısı ve büyüklüğü farklı olabileceğinden yanlış rota çıkmaması için düzeltme fonksiyonu eklenmiştir. Bu durum işlem zamanını artırmaktadır.

Munemoto [15], uyarlanabilir genetik algoritma ile kablolu veya kablosuz ortama uyarlanabilecek bir algoritma geliştirmiştir. Fakat; geliştirilen algoritmanın çalışma süresinin uzun olmasından dolayı büyük ağlar ve gerçek zamanlı uygulamalar için uygun değildir.

Inagaki [16], çoklu yol seçimi için genetik algoritma kullanmıştır. Fakat; yol bulmak için gerekli olan işlem zamanı uzun sürmektedir. Wook'un önerdiği algoritma, Munemoto ve Inagaki'nin önerdikleri algoritmalara göre uygunluk fonksiyonu değerlendirme, işlem zamanı, yol hata oranı açısından daha iyi sonuç vermektedir.

Ayrıca bu alanda yapılan diğer çalışmalar şu şekilde sıralanabilir.

Liu ve Wang [17] tarafından yapılan çalışmada, sinirsel ağların gelişiminde kullanılan en kısa yol problemi için bir algoritma önerilmiştir. Bu algoritmanın gürültülü hopfield sinirsel ağlarında kullanılması öngörülmüştür. Bu algoritmada yapılan işlem, olasılıklı azalan gürültünün devamlı hopfield sinirsel ağlara (HNN-Hopfield Neural Networks) eklenmesidir. Ayrıca enerji fonksiyonu değiştirilmiştir. Bu işlem ise Shortest Path Routing Problem (SPRP) için yapılmıştır. Elde edilen sonuçlarda önerilen yaklaşımın hopfield sinirsel ağlara uygulanan diğer algoritmalardan daha iyi sonuç verdiği görülmüştür.

Gen ve Lin [18] tarafından yapılan çalışmada ise en kısa yol problemini çözmek için rastgele anahtar tabanlı şifreleme ile yeni yol üretme kullanan bir genetik algoritma yaklaşımı sunulmuştur. Ayrıca aritmetik çaprazlama, mutasyon ve geçiş operatörü aracılığıyla bir birleşik algoritma geliştirilmiştir. En kısa yol problemi için yapılan sayısal analiz sonucu elde edilen veriler, önerilen rastgele anahtar tabanlı genetik algoritma yaklaşımının daha yüksek arama kapasitesine sahip olduğunu ve hesaplama zamanını geliştirdiğini göstermiştir.

Lin, Gen ve Cheng [19] tarafından yapılan çalışmada Open Shortest Path First (OSPF) metodundaki en kısa yol problemi için genetik algoritma yaklaşımı sunulmuştur. Önerilen metod, ağda bir yol bulabilmek için öncelik tabanlı şifreleme metodunu kullanmıştır.

Paketler kaynaktan hedefe doğru bir protokol aracılığıyla ağ yolları üzerinden gönderilmektedir. OSPF, en çok kullanılan protokoldür. Buradaki problem ise OSPF link maliyetinin optimizasyonudur. Sayısal analiz sonucunda genetik algoritma yaklaşımının SPRP üzerindeki etkileri değerlendirilmiş ve önerilen yaklaşımın diğer yaklaşımlardan daha iyi olduğu görülmüştür.

Current, Velle ve Cohon [20] tarafından yapılan çalışmada maksimum kapsayan/en kısa yol problemi ve maksimum populasyon/en kısa yol problemi metodları sunulmuştur. Bu işlemi ise mevcut modelin özel durumlarında gerçekleştirmişlerdir. Tüm modeller için olası modifikasyonlar, uzantılar ve uygulamalar sunulmuştur.

Chabrier [21] tarafından yapılan çalışmada araç yönlendirme problemi için kullanılan sütun oluşturma modelinde giriş düzeyde en kısa yol probleminin çözümü için bir yaklaşım sunulmuştur. Buradaki en kötü durum ise bilinen algoritmaların karmaşık olmasıdır. Giriş düzeyindeki sabitler genelde daha basittir. Her müşteri en az bir kere ziyaret edilmelidir. Buradaki iki problem olan giriş yol sabitleriyle beraber veya beraber olmaması durumu için benzer optimal tamsayı çözümleri bulunmaktadır. Bu çalışmada giriş yolları için algoritmalar üzerinde uygulanmak üzere bir tane teorik birçok pratik yaklaşım önerilmiştir. Sonuç olarak arama ağaç yapılarında daha az sınırlama ve budama elde edilmiştir ve bu yaklaşımlar daha önceden açık olan solomon benchmark modelinde bulunan 17 örnek üzerinde tam çözümü bulmuştur.

Misra ve Oommen [22] tarafından yapılan çalışmada dinamik tek kaynaklı en kısa yol problemi için otomata tabanlı öğrenme çözümü sunmaktadır. Tek kaynaklı olasılıklı graf topolojisinde en kısa yolu bulma işlemini içermektedir. Bu topoloji devamlı olasılıksal olarak güncellemeyi kenar ağırlıklarıyla yapmaktadır. Bu algoritma daha önceki algoritmalardan daha verimli olarak çalışmaktadır ve ortalama graf topolojide olasılıksal en kısayol ağacını bulmada kullanılmaktadır. Bu çözüme ulaşırken kenar ağırlıklarında değişiklik olup olmadığına bakmamaktadır. Rastgele yapılan ayarlamalarda da önerilen çözüm en kısa yolu bulmaya yakınlaşmaktadır. Diğer taraftan var olan algoritmalar da değişiklikler gözden geçirilmek zorunda ve her değişiklikte hesaplamalarının tekrar yapılması gerekmektedir.

Cai, Kloks ve Wong [23] tarafından yapılan çalışmada değişik bekleme zamanları için en kısa yol problemi araştırılmıştır. Rastgele bekleme zamanı, sıfır bekleme zamanı, sınırlandırılmış bekleme zamanı olmak üzere üç tane değişik durum incelenmiştir ve sahte polinom algoritmaları bu üç durum için önerilmiştir.

Orda ve Rom [24] tarafından yapılan çalışmada yol maliyetleri ve zamana bağlı gecikmeler göz önüne alınarak en kısa yol problemi araştırılmıştır. Bu problemi çözebilmek için bir algoritma geliştirmişlerdir.

Halpern [25] tarafından yapılan çalışmada zamana bağlı olarak en kısa yol problemi incelenmiştir. Diğer önerilen algoritmaların sadece zamana bağlı gecikmelerin varsayımı üzerine olduğunu belirtmişlerdir. Burada, zamana bağlı gecikmelerde düzenlenerek en kısa yol problemi incelenmiştir.

Cooke ve Halsey [26] tarafından yapılan çalışmada, Bellman algoritması düzenlenerek iki düğüm arasındaki en kısa yol problemi araştırılmıştır.

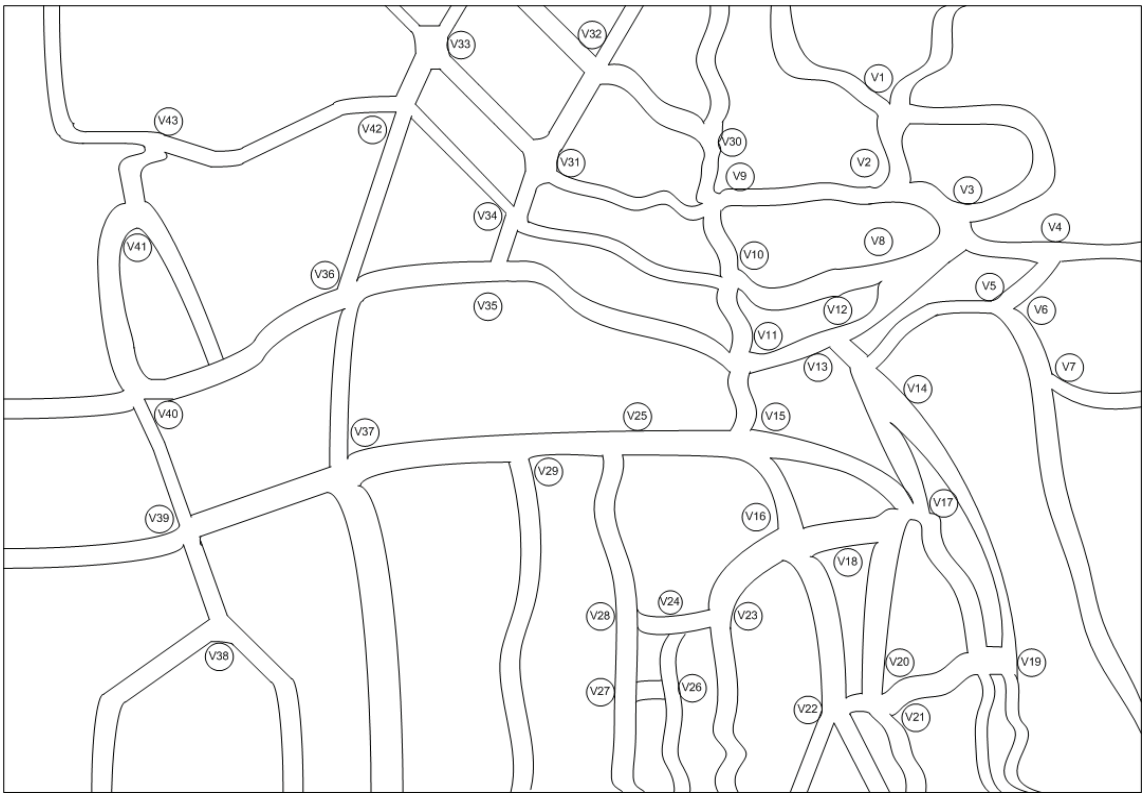
Bu makalede yapılan çalışma, Ankara'da gerçek bir harita üzerinde, Kızılay ve çevresindeki kavşaklar baz alınarak gerçekleştirilmiştir. Bu harita üzerindeki, tüm kavşaklar göz önüne alınmış ve belirli zaman aralıklarında trafiğin yoğun olduğu bölgelerden en kısa sürede istenilen hedefe gidilebilmesi gerçekleştirilmiştir. Bununla birlikte, yolda meydana gelebilecek kaza durumları da göz önüne alınarak yol dinamik olarak sürekli güncellenmiştir.

2. GENETİK ALGORİTMA İLE EN KISA YOL BULMA (FIND THE SHORTEST PATH WITH GENETIC ALGORITHM)

2.1. Trafikte En Kısa Süreli Yolu Bulma Problemi (Find The Shortest Path Problem In Traffic)

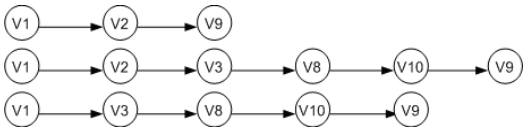
Bir bölgede iş ve okul giriş çıkış saatlerinde trafik yoğunluğu artmaktadır. Belirli saatlerde belli bir bölge trafik açısından yoğun olurken, bunun tersi durum da geçerli olabilmektedir. Örneğin, Türkiye için iş ve okul başlama saati 08:30 ve bitiş saati 17:30'dur. Bu saatlerde ve bu saatlerden 30 dakika önce ve sonra trafik yoğunluğu yaşanmaktadır. Bu yoğunluk tüm yollarda yaşanmamaktadır. Sabah gidiş yolunda yoğunluk fazla, dönüş yolunda yoğunluk az iken, akşam saatlerinde gidiş yolunda yoğunluk az, dönüş yolunda yoğunluk artmaktadır.

Şekil 1'de bu çalışmada kullanılan bölge görülmektedir. Bu şekilde V ile gösterilenler düğüm, aradaki yollar ise ayrıtlardır. Bu şekliyle bir graf yapısını ifade eder. Bir bölgedeyken farklı bir noktaya gitmek için V4 yolu seçildiğinde uzaklık 8 km, V6 yolu seçildiğinde uzaklık 12 km varsayılırsa, V4 yolunu seçmek uygun olacaktır. V6 yolunun uzun olmasına rağmen trafik yoğunluğu az ise, V4 yolunda meydana gelen trafik yoğunluğundan ötürü V6 yolu uygun bir seçim olabilir.

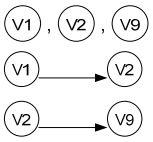


Şekil 1. Geliştirilen uygulamada kullanılan bölge (The region for developed application)

Şekil 1'de gösterildiği gibi V1'den V9'a ulaşmak için bir araç farklı rotalar izleyebilir. Bu rotalara örnek olarak



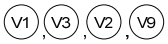
verilebilir. Her geçilen yol için yola gidildiği andaki trafiğe bağlı olarak bir süre harcanacaktır. Bunlar göz önüne alındığında, başlangıç süresi 07:00 seçilirse,



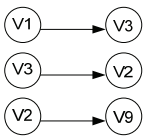
yolu için,

07:00'de 16 dakika

07:16'da 5 dakika alınmıştır.
(Toplam süre = 21 dakika)



yolu için,

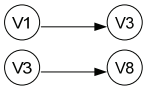


07:00'de 12 dakika

07:12'de 8 dakika

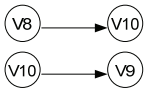
07:20'de 6 dakika alınmıştır.
(Toplam süre = 26 dakika)

yolu için,



07:00'de 12 dakika

07:14'de 2 dakika



07:12'de 2 dakika

07:16'da 2 dakika alınmıştır.
(Toplam süre = 18 dakika)

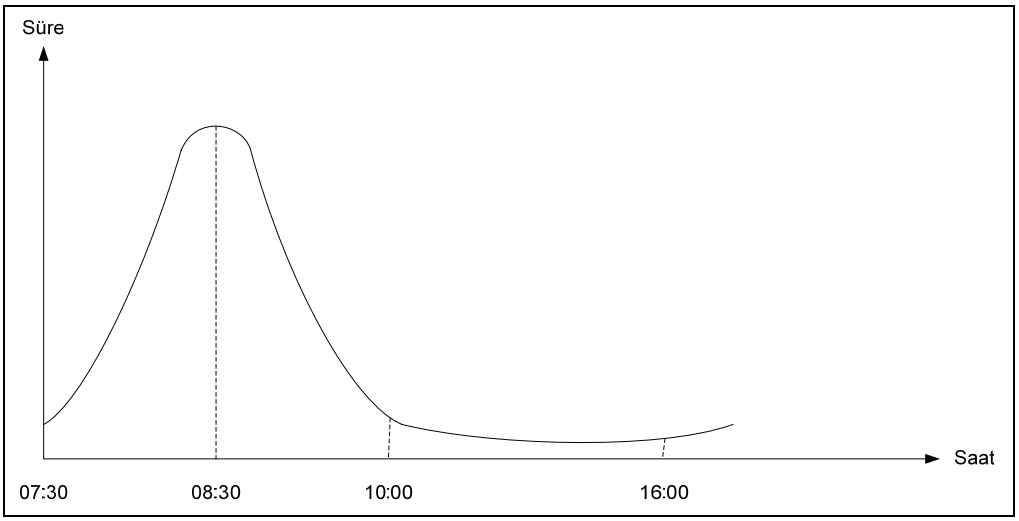
süre gerekecektir.

Örnekte görüldüğü gibi yol uzunluğu fazla olmasına rağmen, 07:00 civarında trafik yoğunluğu bu rotada az olduğu için aracın V1 , V3 , V8 , V10 , V9 yolunu izlemesi gerekmektedir.

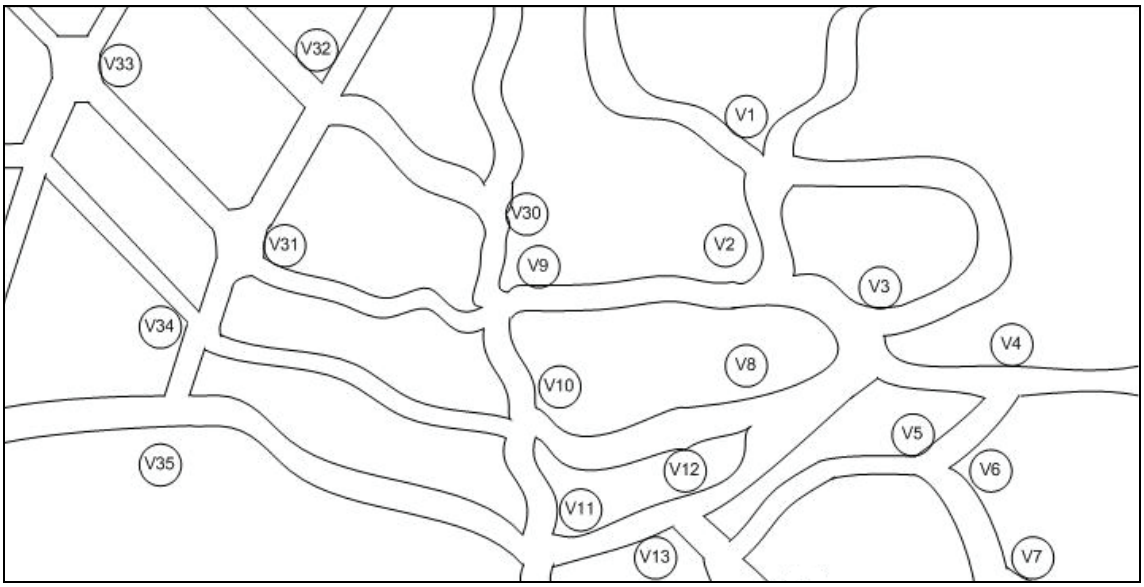
2.2. Genetik Algoritma ile En Kısa Yolun Bulunması (Find The Shortest Path Using Genetic Algorithm)

Bu çalışmada kullanılan Ankara, Kızılay bölgesi için 43 kavşakta, toplam 126 yön için trafik yoğunluk süreleri elde edilmiştir. Örnek olarak Şekil 2'de görüldüğü gibi V1 ve V2 kavşakları arasındaki trafik yoğunluk süresi verilmiştir. Her kavşağa ait 07:00-10:00 / 17:00-20:00 saatleri arasında yoğunluk süreleri depolanmıştır. Diğer saatlerde yoğunluk, ortalama bir süre olarak hesaplanmıştır. Bu saatlerin seçilmesinin amacı, belirlenen bölgedeki iş, okul, çalışma saatlerinde işe veya okula giden insanların bu saatlerde trafiğe çıkmasıdır. Diğer saatlerde yollarda trafik yoğunluğu bulunmamaktadır. Bundan dolayı değerlendirmeye gerek duyulmamıştır. Toplam 45360 veri depolanmıştır.

Yol Arama Algoritması: Başlangıç ve bitiş yolu belirlendikten sonra yol arama algoritması şu şekilde



Şekil 2. V1 ve V2 kavşakları arasındaki trafik yoğunluğu (Traffic density between V1 and V2)



Şekil 3. Yol arama algoritması için örnek harita (Sample map for the path search algorithm)

çalışmaktadır. Başlangıç yolunu kullanarak tek atlamayla gidilebilecek yollar belirlenir. Şekil 3'te yol arama algoritması için örnek harita verilmiştir.

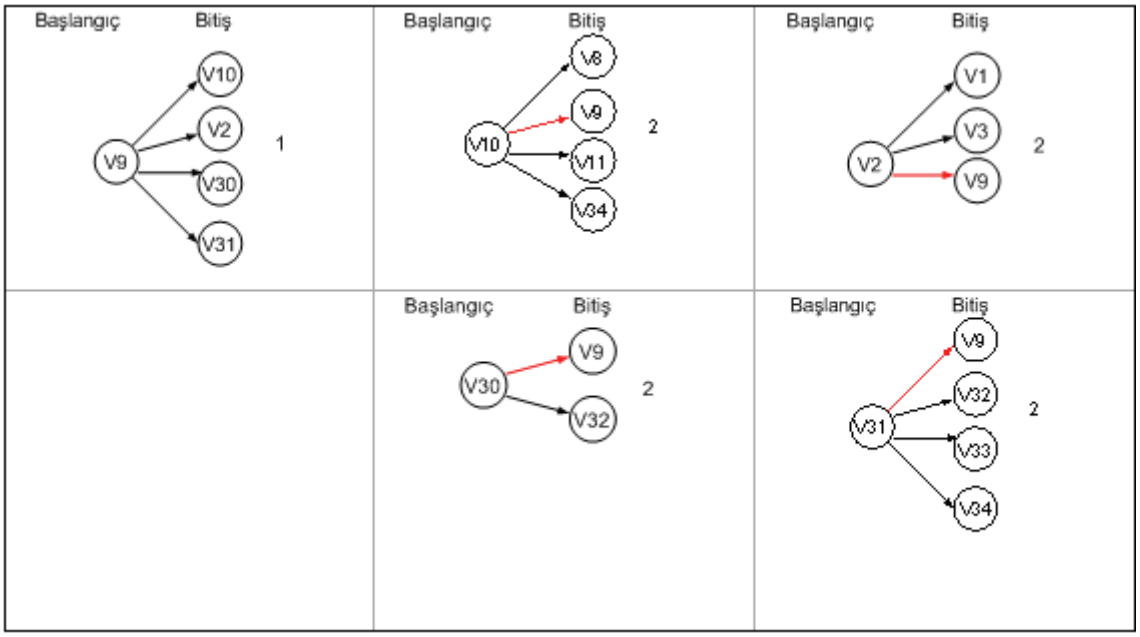
Şekil 3'te görüldüğü gibi başlangıç yolu V9 seçilmişse, birinci atlamada gidilebilecek yerlerin (V2, V10, V30, V31) olduğu ortaya çıkmaktadır. Şekil 4'te yol arama algoritmasının sonuçları verilmektedir.

Şekil 4'te görüldüğü gibi bir sonraki atlama için ise V2, V10, V30, V31'in gidebileceği yollar belirlenmektedir. Görüldüğü gibi her bir atlamada bulunan rota bilgisi artacaktır. Bu işlem bitiş yolu bulunduğunda sonlanacaktır. Bu işlem yapılırken tekrarsızlık (aynı rotanın tekrar bulunması) ve döngüsüzlük (V1 V2 V3 V2 V9) oluşturmak için 2. atlamadan itibaren bitiş yolu başlangıç yoluna eşitse, bulunan rota işleme alınmamıştır. Farklı türden çeşitli yolları bulmak için arama türü bu şekilde seçilmiştir.

Başlangıç Popülasyonu: Belirlenen populasyon sayısı kadar yol arama algoritması çalıştırılarak farklı türden yollar elde edilmektedir. Yol arama algoritmasında tekrar ve döngü olayları engellendiğinden dolayı burada yeni bir kontrole gerek duyulmamıştır. Popülasyon boyutu için farklı sayılarda ugyulama çalıştırılmıştır. En iyi çözümü en kısa sürede elde eden popülashyon boyutu 10 olarak belirlenmiştir. Seçilen kromozomlar, tüm yol alternatifleri arasından rasgele seçilmektedir. Tablo 1'de başlangıç popülasyonuna ait kromozomlar ve başlangıç ile bitiş saatleri verilmiştir.

Tablo 1'de görüldüğü gibi 10 tane farklı yol bulunmuştur. Sağdaki iki sütunda ise başlangıç saati ve bitiş saati görülmektedir.

Uygunluk Fonksiyonu (UF): Uygunluk fonksiyonunun belirlenmesi, genetik algoritmanın etkinliğini en öenmli derecede etkileyen



Şekil 4. Yol Arama Algoritması sonuçları (Results of path search algorithm)

Tablo 1. Başlangıç Popülasyonuna ait kromozomlar (Chromosomes of initial population)

No	Kromozomlar										Başlama saati	Bitiş saati
1	V1	V2	V9								07:00	07:50
2	V1	V3	V2	V9							07:00	07:51
3	V1	V3	V8	V10	V9						07:00	07:18
4	V1	V2	V3	V8	V10	V9					07:00	07:32
5	V1	V3	V8	V10	V34	V31	V9				07:00	07:23
6	V1	V2	V3	V8	V10	V34	V31	V9			07:00	07:40
7	V1	V3	V4	V5	V13	V12	V11	V10	V9		07:00	07:29
8	V1	V3	V8	V10	V34	V31	V32	V30	V9		07:00	07:29
9	V1	V2	V3	V4	V5	V13	V12	V11	V10	V9	07:00	07:32
10	V1	V2	V3	V8	V10	V34	V31	V32	V30	V9	07:00	07:50

faktördür[15]. Probleme tam uygun olarak oluşturulmamış bir uygunluk fonksiyonu, çalışma süresini uzatmakta hatta çözüme hiçbir zaman ulaşılamamasına neden olabilmektedir. Uygunluk fonksiyonunu belirlemek için öncelikle bitiş süresinden başlangıç süresi çıkarılmıştır. Bu değer ne kadar az ise çözüme de o kadar uygundur. Uygunluk fonksiyonu aşağıdaki gibi belirlenmiştir.

$$UF = 1 \quad / \sum_{i=gecilentümyollar} Yolmaliyeti_i \qquad (1)$$

Elitizm: Elitizm, genetik algoritmada seçme işlemi için kullanılır. Elit birey alınmazsa yeni jenerasyonun en iyi bireyi bir önceki jenerasyonun en iyi bireyinden daha kötü olabilir. Elit birey sayısı arttıkça da çözümdeki çeşitlilik azalır. Bundan dolayı belirli sayıdaki en iyi birey hiçbir işleme tabi tutulmadan doğrudan yeni jenerasyona aktarılması gerekir.

Bu çalışmada, yoğunluğu azaltma ve trafikte geçen zamanı en aza indirgemek için süre bakımından en düşük olan 2 birey yeni popülasyona aktarılmıştır.

Tablo 2. Seçme İşlemi (Selection)

Sıra	Başlama Saati	Bitiş Saati	Süre	Toplam / Süre	Kriter Toplamı
1	07:00	07:50	50	7	7
2	07:00	07:51	51	7	14
3	07:00	07:18	18	20	34
4	07:00	07:32	32	11	45
5	07:00	07:23	23	15	60
6	07:00	07:40	40	9	69
7	07:00	07:29	29	12	81
8	07:00	07:29	29	12	93
9	07:00	07:32	32	11	104
10	07:00	07:50	50	7	**111**
		Süre Toplamı= 354			

Seçme: Seçme işlemi için rulet tekerleği yöntemi uygulanmıştır. Rulet tekerleğinde her bireyin çözüme uygunluk derecesi arttıkça yeni popülasyona aktarılma şansı artar. Populasyonda bulunan kromozomların başlangıç ve bitiş zamanı belirlendikten sonra bu değerler arasındaki zaman farkı hesaplanır. Tüm farkların toplamı her bir farka bölünür. Ardından populasyondaki ilk veriden başlayarak, son veriye kadar değerler toplanır. Tablo 2'de seçme işlemi için örnek hesaplama verilmiştir.

Kriter hesaplandıktan sonra, 0' dan 111'e kadar rasgele bir sayı seçilir. Ulaşılmaya çalışılan verinin aralığı yüksek olduğundan o verinin gelme ihtimalide yüksektir. Bu şekilde birey seçme işlemi gerçekleşir.

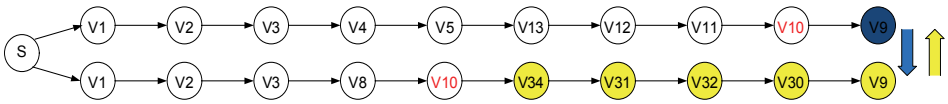
Çaprazlama: Çaprazlama, iki kromozomun birbirleri arasında gen alışverişinde bulunup iki yeni kromozom oluşturmasıdır. Genetik algoritmadaki en önemli operatörlerinden birisidir. Önerilen algoritmada seçilen iki birey arasında, başlangıç ve bitiş yolları hariç olmak üzere Şekil 5'te görüldüğü gibi ortak bir gen bulunur. Bu şekilde çaprazlama işlemi gerçekleşir.

Öncelikle birinci birey seçilir. Ardından birinci bireye eşit olmayan ikinci birey seçilir. Seçilen ikinci birey, birinci bireye eşit olduğu sürece bu işlem tekrarlanır. Seçilen iki birey arasında ortak gen aranır. Ortak gen bulunamadığı sürece ikinci birey seçimi tekrarlanır. Ortak gen bulunduğunda, iki birey arasında değiş-tokuş işlemi yapılarak oluşan Şekil 6'da görüldüğü gibi yeni iki birey yeni popülasyona aktarılır.

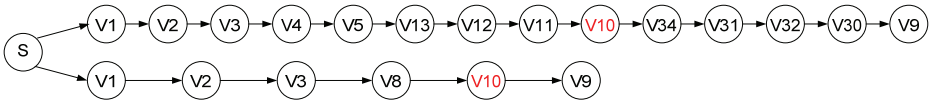
Algoritmaya ait kaba kod aşağıda verilmiştir.

```
repeat
  ilksecim:=secim();
  do
    ikincisecim:=secim();
  while (ilksecim==ikincisecim)

  for (i,j)
  begin
    ortakgen(ilksecim[],ikincisecim[]);
  end;

  if (sonuc==1)
    degistokus(ilksecim,ikincisecim);
    yenipopulasyonaaktar(yenibir);
    yenipopulasyonaaktar(yeniiki);
  else
    sonuc==0;
until sonuc='1';
```

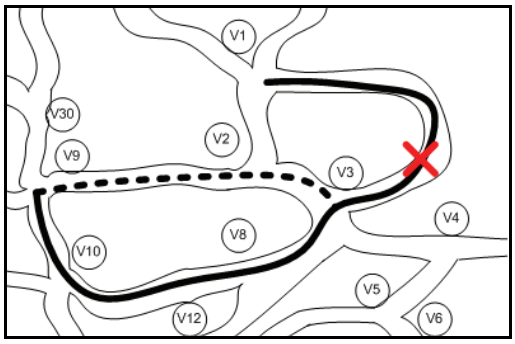
Kaza Oluşturma: Trafik yoğunluğu ve yol maliyeti olarak en iyi yol genetik algoritmayla bulunmasına rağmen seçilen yolda meydana gelen bir kaza trafiği aksatabilir ve güzergahın değişmesine sebep olabilir. Bu bölümde de seçilen yolda kaza meydana geldiğinde yapılan işlemler örnek bir senaryo kullanılarak açıklanmıştır. Şekil 7'de kaza oluşturma verilmiştir.



Şekil 5. Çaprazlama işlemi I (Crossover I)



Şekil 6. Çaprazlama işlemi II (Crossover II)



Şekil 7. Kaza oluşturma (Accident occurs)

Şekil 7'de başlangıç yolu V1, bitiş yolu V9 için en kısa süreli yolun V1 – V3 – V8 – V10 – V9 olacağı görülmektedir. Bu yollar arasında kaza meydana gelebilir. V1 ve V3 arasında bir kaza oluşması durumunda aracın 5 dakika gecikeceğini varsayalım. Yani V8'e tahmin edilen süreden 5 dakika geç gelmiş olacaktır. Bu süre zarfında V8-V10-V9 yolu yoğunlaşmış olabilir. Bunun için araç V8'de iken V9'a ulaşmak için tekrar genetik algoritma ile yol arar. Gerçekleştirilen algoritmada 3 saniyede araç yeni rotasını bulabilmektedir. Artık aracın rotası yeni bulduğu yoldur. V9'a yeni bulduğu rota üzerinden devam eder. Algoritmaya ait kod aşağıda verilmiştir.

```
kaza=false;
rota=ilkyol();
do
  rota=sonraki_yol();
  if kaza=true break;
while kaza=false;

if kaza=true
yeni_yol_ara();
```

3. ÖNERİLEN ALGORITMANIN TASARIMI (DESIGN OF THE PROPOSED ALGORITHM)

Önerilen sistemde, veritabanı olarak MSSQL 2005, programlama dili olarak Borland Delphi 7 kullanılmıştır. Tüm benzetimler Pentium M 2.0GHz işlemciye sahip bilgisayarda gerçekleştirilmiştir. Oluşturulan tablolar kısaca aşağıda açıklanmıştır.

Rota Tablosu, Gerçekleştirilen yazılımda simülasyon ortamı olduğundan dolayı her kavşak için rota bilgisine ihtiyaç duyulmaktadır. Rota tablosunda bir noktadan, sonraki bir noktaya gitmek için gereken x,y koordinatları tutulmaktadır. Toplam 349 veri bulunmaktadır.

Graflar Tablosu, Bu tabloda, 43 tane noktaya ait x ve y koordinatları tutulmaktadır.

Süre Tablosu, Bu tabloda, her bir iki nokta arasında 07:00 / 10:00 ve 17:00 / 20:00 saatlerindeki yoğunluklar tutulmaktadır. Toplam 126 yön için (07:00 / 10:00 arası 3 saat, 17:00 / 20:00 arası 3 saat, 3+3=6 saat= 360 dakika*126 yön) 45360 veri içermektedir. Muhtemel yol aranırken hangi saatlerde ne kadar yoğunluğun olduğu bu tablo sayesinde öğrenilmektedir.

Yollar Tablosu, Bu tablo program çalıştırıldıktan sonra dolmaktadır. Yol arama esnasında veriler buraya işlenir. Eğer gidilmek isenilen yol bulunmuşsa tamam alanı 1 değerini alır, aksi takdirde 0'dır. Başlangıç popülasyonu sayısı kadar tamam alanında 1 değeri varsa işlem sonlanmaktadır.

Populasyon Tablosu, Başlangıç populasyonu oluştuktan sonraki değerler bu tabloya kaydedilmektedir. Ayrıca, her bir yeni populasyondaki değerler bu tabloda güncellenerek işlenmektedir.

Pop_Sureler Tablosu, Buradaki sayı alanıyla populasyon tablosundaki sayı alanı ortaktır. Populasyon tablosunda bir yol bilgisi olmasına karşılık, bu tabloda da o yola ait başlangıç ve bitiş süreleri tutulmaktadır. Fark, fonksiyon ve kriter ise rulet tekerliği ile seçim yapmak için oluşturulmuştur.

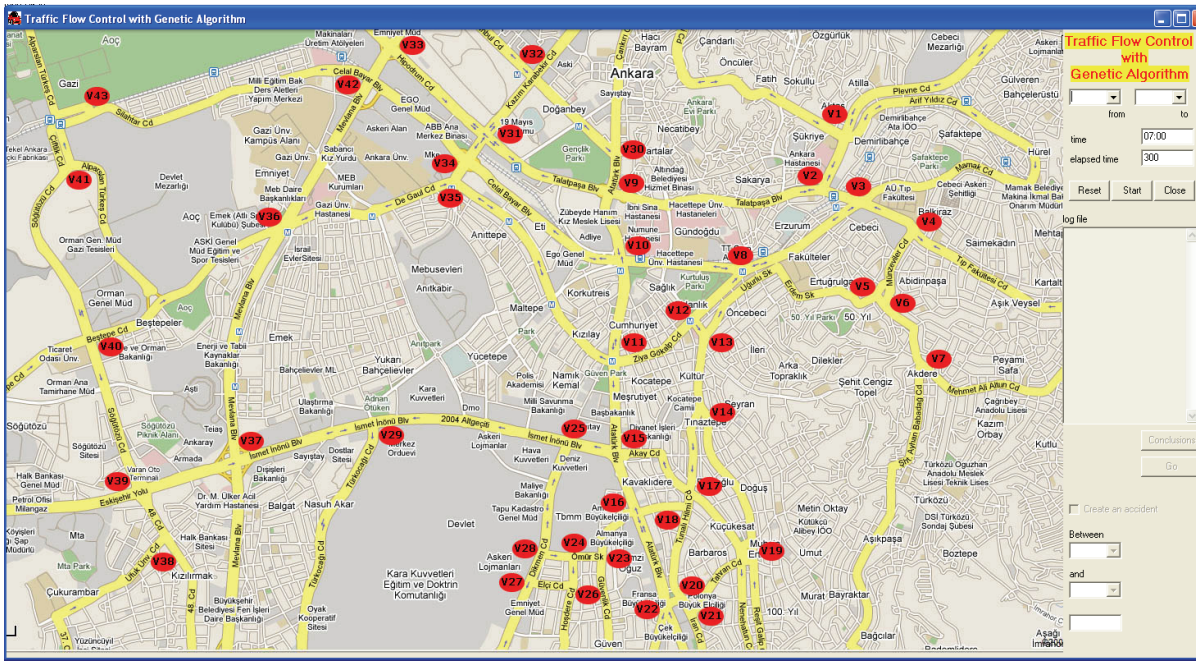
İkamet Tablosu, İkamet tablosunda ise, belirtilen jenerasyon sayısı bittiğinde en iyi olan yolu düğüm düğüm kaydeden tablodur. Programda arabanın hareketi bu tabloyu kullanarak gitmektedir.

Geliştirilen simülasyona ait ekran görüntüsü Şekil 8'de verilmiştir.

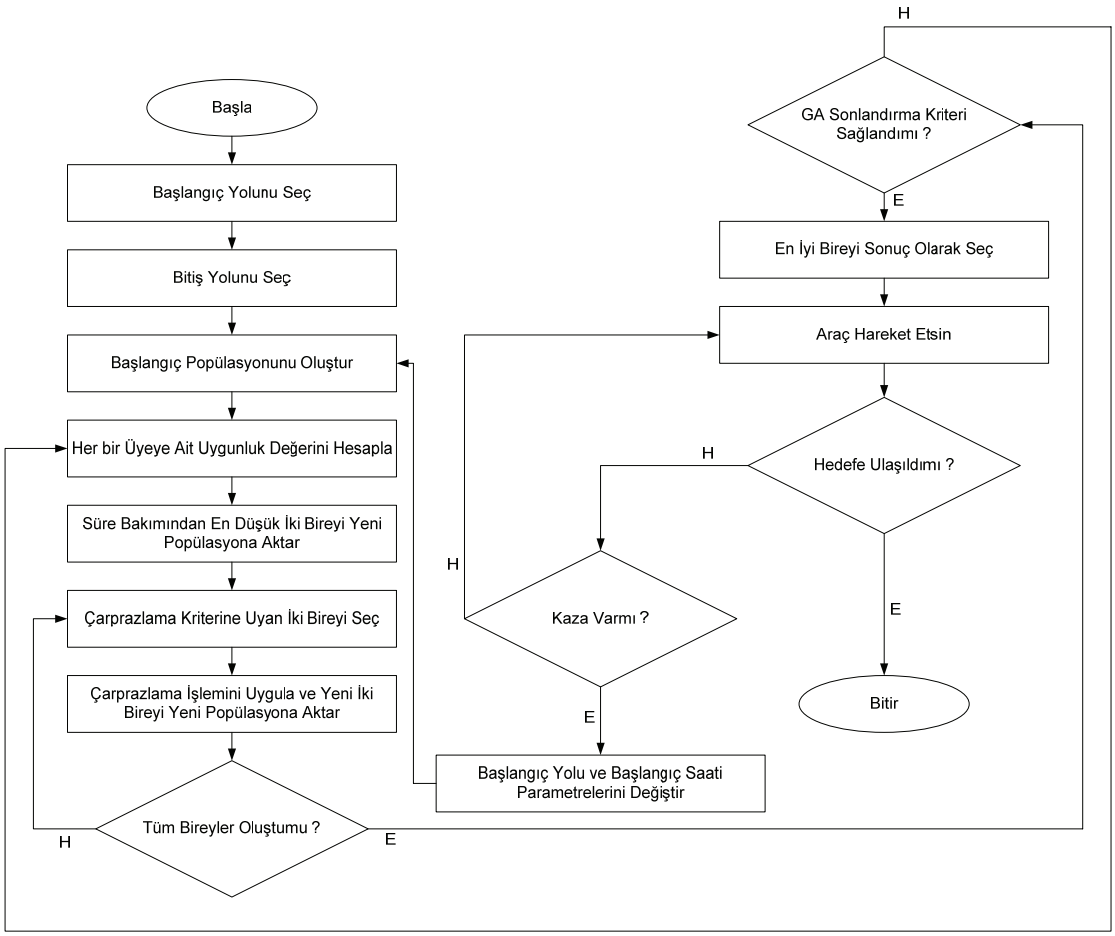
Şekil 8'de görüldüğü gibi yapılan çalışma Ankara'da gerçek bir harita üzerinde, Kızılay ve çevresindeki kavşaklar baz alınarak gerçekleştirilmiştir. Şekil 6'da görülen tüm kavşaklar göz önüne alınmış ve belirli zaman aralıklarında trafiğin yoğun olduğu bölgelerden en kısa sürede istenilen hedefe gidilebilmesi gerçekleştirilmiştir.

Uygulama yazılımına ait akış şeması Şekil 9'da verilmiştir.

Şekil 9'da verilen akış diyagramında, geliştirilen yazılımın hangi aşamalardan geçeceği detaylı olarak gösterilmiştir. Program ilk önce başlangıç yolunu seçmekte, sonra Bitiş yolunu seçmektedir. Bu iki değer seçildikten sonra programın çözümlemeyi gerçekleştireceği saat belirlenmekte ve bu işlemden sonra ilk popülasyon oluşturulmaktadır. Oluşturulan bu popülasyonların uygunluk değerleri hesaplanır. Daha sonra süre bakımından en düşük iki birey yeni popülasyona dahil edilir. Popülasyonları daha iyi hale getirmek için çaprazlama kriterine uyacak bireyler seçilir ve çaprazlama işlemi sonucunda ortaya çıkan yeni 2 birey yeni popülasyona aktarılır. Bu şekilde tüm bireyler oluşturulana kadar çaprazlama işlemi gerçekleştirilir. Eğer tüm bireyler oluşturulduysa bitirme kriteri göz önünde bulundurulur. Bitirme

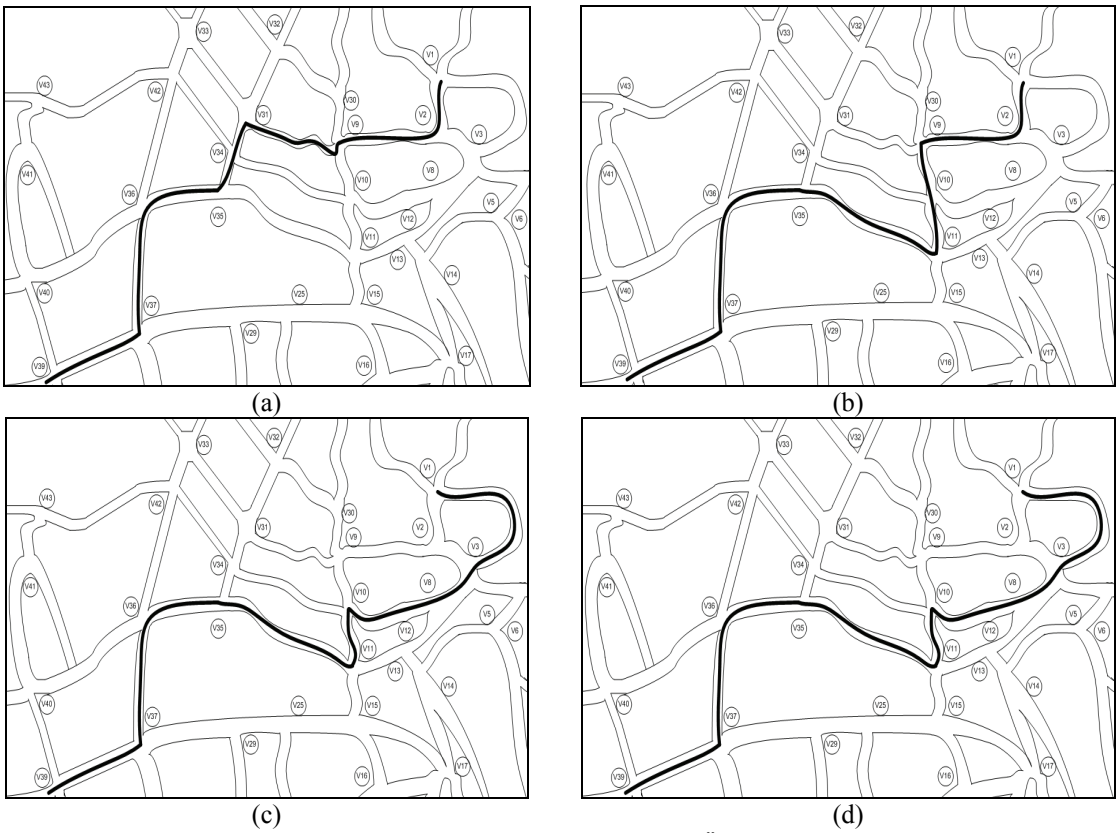


Şekil 8. Geliştirilen Simülasyon Yazılımının Ekran Görüntüsü (Screenshot of the Developed Simulation Software)



Şekil 9. Önerilen algoritmanın akış diyagramı (Flow chart of the proposed algorithm)

kriteri sağlanmadıysa herbir üyeye ait uygunluk değeri hesaplanır ve yukarıda anlatılan ve şekilde gösterilen işlemler devam eder. Bitirme kriteri sağlandıysa en iyi birey sonuç olarak seçilir ve araç harekete başlar. Yolda kaza yoksa harekete devam edilir ve hedefe ulaşılır. Yolda kaza var ise yeniden başlangıçtaki işlemler yapılarak başlangıç popülasyonu oluşturulur ve aynı işlemler tekrarlanır.



Şekil 10. (a) Wook'un algoritması, (b) Dijkstra algoritması, (c) Önerilen algoritma, (d) Optimum yol
((a) Wook's algorithm, (b) Dijkstra algorithm, (c) Proposed algorithm, (d) Optimum path)

4. BENZETİM SONUÇLARI (SIMULATION RESULTS)

Bu bölümde önerilen genetik algoritma Dijkstra ve Wook'un algoritmasıyla karşılaştırılmıştır. Dijkstra ve Wook'un algoritmaları başlangıçta yolun bulunması için kullanılmış ve her düğümde tekrar çalıştırılarak yol güncellenmiştir. Bu şekilde çalıştırılarak yoldaki dinamik değişime göre çalışması sağlanmıştır. Wook'un algoritması, Munetomo'nun algoritması [15] ile Inagaki'nin algoritmasından [16] daha iyi sonuç verdiği için bu algoritmalarla karşılaştırmaya gerek duyulmamıştır. Karşılaştırma sonucu Şekil 10'da verilmiştir.

Çözümlenen problemde toplam 43 adet nokta bulunmaktadır. Populasyon sayısı 10 olarak seçilmiştir. Trafik yoğunluğunun olmadığı saatlerde (10:00-16:00 / 20:00-07:00) yukarıda belirtilen algoritmalar aynı optimum yolu seçmektedir. Fakat; bu saatler dışında trafik yoğunluğu olduğunda önerilen algoritma daha iyi sonuç vermektedir. V1 noktasından V39 noktasına, saat 07:00'de araç hareket ettiğinde sonuçlar Şekil 8'deki gibi olmaktadır.

Koyu olan çizgi aracın hedefe ulaşırken kullandığı rotayı belirtmektedir. Benzetim sonucunda hedefe ulaşmak için optimum süre 8 dakikadır. Bu süreyi önerilen algoritma bulmuştur. Optimum sürenin bulunduğu yolun maliyeti ise 10 km'dir. Yol Dijkstra algoritmasıyla arandığında geçen süre 14 dakika ve yolun maliyeti 7 km'dir. Wook'un algoritmasıyla arandığında geçen süre 11 dakika ve yolun maliyeti 9 km'dir.

Önerilen algoritmada birincil amaç en kısa süreli yolu bulmak olduğu için, yol uzunluğu en fazla çıkmaktadır. Bu eksik olarak görülse de aşağıda verilen açıklamalar bu olayı kabul etmemektedir. Araç, normal hızda seyir halindeyken daha az benzin yakmaktadır ve havayı daha az kirletmektedir. Yoğun olan yollarda ise aracın dur-kalk şeklinde hareket etmesiyle hem benzin hem de egzoz dumanları havayı daha fazla kirletmektedir. Tablo 3'te ilk satırda başlangıç yolları ve sol bölümünde de zaman aralığı verilmiştir. Parantez içindeki değer iki yol arasındaki km cinsinden uzunluğu belirtirken, yanındaki sayı da dakika cinsinden yolun ne kadar zaman aldığını göstermektedir. Örneğin, V10'dan V34'e 17:00-17:10 arasında gidildiğinde uzunluğu 2 km olan yol 3 dakika sürmektedir.

Tablo 3 esas alınarak aşağıdaki hesaplamalar gerçekleştirilmiştir. Tablo 3'ten başlangıç yolu V43, Bitiş Yolu V11, saat: 17:20 seçildiğinde muhtemel yollar Tablo 4'de görüldüğü gibi olacaktır.

Tablo 3. Kavşaklara ait başlangıç/zaman değerleri (Initial-time values of junctions)

Başlangıç – Zaman	V10	V11	V15	V25	V29	V34	V35	V36	V37	V42	V43
17:00 - 17.10	V11 (1) 1 V34 (2) 3	V10 (1) 4 V15 (1) 3 V35 (4,5) 22	V11 (1) 3 V25 (0,5) 4	V15 (0,5) 4 V29 (2,5) 8	V25 (2,5) 16 V37 (2) 7	V10 (2) 3 V35 (0,2) 1 V42 (0,8) 2	V11 (4,5) 18 V34 (0,2) 1 V36 (3) 6	V35 (3) 7 V37 (3) 12 V42 (2) 7	V29 (2) 9 V36 (3) 12	V34 (0,8) 2 V36 (2) 7 V43 (4) 10	V42 (4) 6
17:10 - 17.20	V11 (1) 1 V34 (2) 4	V10 (1) 4 V15 (1) 4 V35 (4,5) 23	V11 (1) 4 V25 (0,5) 5	V15 (0,5) 5 V29 (2,5) 9	V25 (2,5) 17 V37 (2) 8	V10 (2) 4 V35 (0,2) 1 V42 (0,8) 2	V11 (4,5) 19 V34 (0,2) 1 V36 (3) 7	V35 (3) 8 V37 (3) 13 V42 (2) 8	V29 (2) 10 V36 (3) 13	V34 (0,8) 5 V36 (2) 8 V43 (4) 11	V42 (4) 7
17:20 - 17.30	V11 (1) 1 V34 (2) 5	V10 (1) 5 V15 (1) 4 V35 (4,5) 24	V11 (1) 4 V25 (0,5) 6	V15 (0,5) 6 V29 (2,5) 10	V25 (2,5) 18 V37 (2) 8	V10 (2) 5 V35 (0,2) 2 V42 (0,8) 2	V11 (4,5) 20 V34 (0,2) 2 V36 (3) 8	V35 (3) 9 V37 (3) 15 V42 (2) 9	V29 (2) 11 V36 (3) 15	V34 (0,8) 11 V36 (2) 6 V43 (4) 12	V42 (4) 8
17:30 - 17.40	V11 (1) 2 V34 (2) 5	V10 (1) 5 V15 (1) 5 V35 (4,5) 25	V11 (1) 5 V25 (0,5) 6	V15 (0,5) 7 V29 (2,5) 10	V25 (2,5) 18 V37 (2) 8	V10 (2) 11 V35 (0,2) 2 V42 (0,8) 3	V11 (4,5) 21 V34 (0,2) 2 V36 (3) 8	V35 (3) 6 V37 (3) 15 V42 (2) 9	V29 (2) 10 V36 (3) 15	V34 (0,8) 16 V36 (2) 9 V43 (4) 12	V42 (4) 8
17:40 - 17.50	V11 (1) 2 V34 (2) 5	V10 (1) 5 V15 (1) 5 V35 (4,5) 25	V11 (1) 5 V25 (0,5) 5	V15 (0,5) 7 V29 (2,5) 8	V25 (2,5) 18 V37 (2) 7	V10 (2) 3 V35 (0,2) 2 V42 (0,8) 3	V11 (4,5) 20 V34 (0,2) 11 V36 (3) 8	V35 (3) 10 V37 (3) 15 V42 (2) 10	V29 (2) 11 V36 (3) 15	V34 (0,8) 10 V36 (2) 10 V43 (4) 13	V42 (4) 8
17:50 - 18.00	V11 (1) 6 V34 (2) 4	V10 (1) 4 V15 (1) 4 V35 (4,5) 24	V11 (1) 4 V25 (0,5) 4	V15 (0,5) 7 V29 (2,5) 9	V25 (2,5) 17 V37 (2) 9	V10 (2) 4 V35 (0,2) 1 V42 (0,8) 3	V11 (4,5) 20 V34 (0,2) 1 V36 (3) 7	V35 (3) 9 V37 (3) 13 V42 (2) 9	V29 (2) 10 V36 (3) 14	V34 (0,8) 8 V36 (2) 9 V43 (4) 12	V42 (4) 9
18:00 - 18.10	V11 (1) 1 V34 (2) 3	V10 (1) 4 V15 (1) 4 V35 (4,5) 23	V11 (1) 4 V25 (0,5) 4	V15 (0,5) 6 V29 (2,5) 9	V25 (2,5) 14 V37 (2) 7	V10 (2) 3 V35 (0,2) 1 V42 (0,8) 2	V11 (4,5) 19 V34 (0,2) 1 V36 (3) 6	V35 (3) 8 V37 (3) 14 V42 (2) 8	V29 (2) 9 V36 (3) 13	V34 (0,8) 6 V36 (2) 8 V43 (4) 11	V42 (4) 8
18:10 - 18.20	V11 (1) 1 V34 (2) 3	V10 (1) 3 V15 (1) 3 V35 (4,5) 23	V11 (1) 3 V25 (0,5) 3	V15 (0,5) 5 V29 (2,5) 8	V25 (2,5) 14 V37 (2) 6	V10 (2) 3 V35 (0,2) 1 V42 (0,8) 2	V11 (4,5) 18 V34 (0,2) 1 V36 (3) 5	V35 (3) 7 V37 (3) 13 V42 (2) 7	V29 (2) 9 V36 (3) 12	V34 (0,8) 2 V36 (2) 7 V43 (4) 10	V42 (4) 7

Tablo 4. Muhtemel yollar (Probable paths)

1)	V43	V42	V34	V10	V11		
V43	V42	=	17:20		17:28	4	km
V42	V34	=	17:28		17:39	0,8	km
V34	V10	=	17:39		17:50	2	km
V10	V11	=	17:50		**17:56**	1	km
						7,8	*km*

2)	V43	V42	V34	V35	V11		
V43	V42	=	17:20		17:28	4	km
V42	V34	=	17:28		17:39	0,8	km
V34	V35	=	17:39		17:41	0,2	km
V35	V11	=	17:41		**18:01**	4,5	km
						9,5	*km*

3)	V43	V42	V36	V37	V29	V25	V15	V11
V43	V42	=	17:20		17:28		4	km
V42	V36	=	17:28		17:34		2	km
V36	V37	=	17:34		17:49		3	km
V37	V29	=	17:49		18:00		2	km
V29	V25	=	18:00		18:14		2,5	km
V25	V15	=	18:14		18:19		0,5	km
V15	V11	=	18:19		**18:22**		1	km
							15	*km*

4)	V43	V42	V36	V35	V11		
V43	V42	=	17:20		17:28	4	km
V42	V36	=	17:28		17:34	2	km
V36	V35	=	17:34		17:40	3	km
V35	V11	=	17:40		**18:00**	4,5	km
						13,5	*km*

5)	V43	V42	V36	V35	V34	V10	V11	
V43	V42	=	17:20		17:28		4	km
V42	V36	=	17:28		17:34		2	km
V36	V35	=	17:34		17:40		3	km
V35	V34	=	17:40		17:41		0,2	km
V34	V10	=	17:41		17:44		2	km
V10	V11	=	17:44		**17:46**		1	km
							12,2	*km*

Bulunan 5 rotaya ait toplam yol uzunluğu italik olarak, 17:20'de başlangıç noktasından hareket eden aracın saat kaçta bitiş noktasına vardığı ise koyu olarak gösterilmiştir. Tablolardan görüldüğü gibi en kısa süreli yol, 17:46'da sonlanan V43-V42-V36-V35-V34-V10-V11 yoludur. Önerilen algoritma sona ulaşmak için 5. rotayı tercih etmiştir. Dijkstra algoritması 1. rotayı yolu tercih ederken, Wook'un algoritması 2. rotayı tercih etmiştir.

Ayrıca geliştirilen algoritma ile diğer algoritmalar süreye görede karşılaştırılmışlardır. Her bir algoritmaya ait, yol seçimi yapılırken geçen süre Tablo 5'te verilmiştir.

Önerilen algoritma ve diğer algoritmalar her yeni kavşakta kalan yol ve mevcut dinamik durum için yeniden çalıştırıldığında elde edilen süreler karşılaştırılmıştır. Geliştirilen algoritma diğer algoritmalara göre daha kısa sürede yol seçme işlemini yapmaktadır.

Tablo 5. Algoritmaların çalışma süreleri (Processing time for the algorithms)

Atlama Sayısı	Yollar	Wook'un algoritması için süre (s)	Dijkstra algoritması için süre (s)	Önerilen algoritma için süre (s)
1	V39 – V37	0,9	1,1	0,6
2	V37 – V36	0,7	0,9	0,5
3	V36 – V35	0,7	0,8	0,4
4	V35 – V34	0,7		
	V35 – V11		0,9	0,4
5	V34 – V31	0,5		
	V11 – V10		0,6	0,3
6	V31 – V9	0,5		
	V10 – V9		0,7	
	V10 – V8			0,4
7	V9 – V2	0,5	0,7	
	V8 – V3			0,3
8	V2 – V1	0,5	0,6	
	V3 – V1			0,3
	Toplam Süre	**5,0**	**6,3**	**3,2**

5. SONUÇLAR (CONCLUSIONS)

Bu makalede en kısa yol probleminin çözümü için genetik algoritma tabanlı bir algoritma geliştirilmiştir. Geliştirilen algoritma zamana bağlı dinamik değişen yol maliyetlerine sahip 43 düğümlü bir graf üzerinde en kısa yol probleminin çözümü için uygulanmıştır. Gerçek bir uygulama olması için bir bölgenin trafik akışı modellenmiştir. Geliştirilen algoritmayla elde edilen sonuçlar literatürde yer alan Dijkstra ve Wook algoritmalarıyla karşılaştırılmıştır. Benzetim sonuçlarından geliştirilen algoritmanın zamana bağlı dinamik değişen yol maliyetleri kullanıldığında daha kısa sürede daha düşük maliyetli bir yol bulunduğunu göstermiştir.

KAYNAKLAR (REFERENCES)

1. Dijkstra, E.W., "A Note on Two Problems in Connection with Graphs", **Numerische Mathematik**, Vol 1, 269–271, 1959.
2. Dantzig, G.B., "On the Shortest Route Through a Network", **Management Science,** 187–190, 1960.
3. Floyd, R.W., "Algorithm 97: Shortest Path", **Communications of the ACM 5**, 1962.
4. Beasley, D., Bull, D.R., and Martin, R.R., 1993a. "An Overview of Genetic Algorithms: Part 1", **Fundamentals. University Computing,** Vol 15(2), 58–69, UK, 1993.
5. Beasley, D., Bull, D.R., and Martin, R.R., 1993b. "An Overview of Genetic Algorithms: Part 2", **Research Topics University Computing,** Vol 15(4), 170–181, UK, 1993.
6. Bingul, Z., Sekmen, A.S. and Zein, S., "An Application of Multi-Dimensional Optimization Problems Using Genetic Algorithms", **Proceedings of the IASTED International Conference Intelligent Systems and Control,** Santa Barbara, CA, USA, 1999.
7. Bingul, Z., Sekmen, A.S. and Zein, S., "Genetic Algorithms Applied to Real Time Multi-objective Optimization Problems", **IEEE SoutheastCon 2000 Conference,** Nashville, TN, USA, 2000.
8. Drezner, Z. and Wesolowsky, G.O., "Network Design: Selection and Design of Links and Facility Location", **Transportation Research Part A: Policy and Practice,** Vol 37, No 3, 241–256, 2003.
9. Xiaoyu J., "Models and Algorithm for Stochastic Shortest Path Problem", **Applied Mathematics and Computation**, Vol 170, No 1, 503–514, 2005.
10. Daniele C., Martine L., Martha Salazar N., "Reduction Approaches for Robust Shortest Path Problems", **Computers &Operations Research**, Vol 38, No 11, 1610–1619, 2011.
11. Bellman, E., "On a Routing Problem", **Appl. Math.,** Vol 16, 87–90, 1958.
12. Dijkstra, E.W., "A Note on Two Problems in Connection with Graphs", **Numer. Math. 1,** Vol 1, 269–271, 1959.
13. Dreyfus, S., "An Appraisal of Some Shortest Path Algorithms", **Oper. Res.,** Vol 17, No 3, 395–412, 1969.
14. Wook, C., Ramakrishna, R.S., "A Genetic Algorithm for Shortest Path Routing Problem and the Sizing of Populations", **IEEE Transactions on Evolutionary Computation**, Vol 6, No 6, 2002.
15. Munemoto, M., Takai, Y., Sato, Y., "A Migration Scheme for the Genetic Adaptive Routing

Algorithm," **IEEE Int. Conf. Systems, Man, and Cybernetics**, 2774–2779, 1998.

16. Inagaki, J., Haseyama, M., Kitajima, H., "A Genetic Algorithm for Determining Multiple Routes and Its Applications," **IEEE Int. Symp. Circuits and Systems**, 137–140, 1999.
17. Liu, W., Wang, L.P., "Solving the Shortest Path Routing Problem Using Noisy Hopfield Neural Networks", **WRI International Conference on Communications and Mobile Computing: CMC 2009**, Vol 2, 299-302, 2009.
18. Gen, M., Lin, L., "A New Approach for Shortest Path Routing Problem by Random Key-Based GA", **Gecco 2006: Genetic and Evolutionary Computation Conference**, Vol 1 and 2 , 1411–1412, 2006.
19. Lin, L., Gen, M., Cheng, R.W.," Priority-Based Genetic Algorithm for Shortest Path Routing Problem in OSPF", **Proceedings of the Third International Conference on Information and Management Sciences**, Vol 3, 411–418, 2004.
20. Current, J.R., Velle, C.S.R., Cohon, J.L., "The Maximum Covering/Shortest Path Problem: A Multiobjective Network Design and Routing Formulation", **European Journal of Operational Research**, Vol 21(2), 189–199, 1985.
21. Chabrier, A., "Vehicle Routing Problem with Elementary Shortest Path Based Column Generation", **Computers & Operations Research**, Vol 33(10), 2972–2990, 2006.
22. Misra, S., Oommen, B.J., "Dynamic Algorithms for the Shortest Path Routing Problem: Learning Automata-Based Solutions", **IEEE Transactions On Systems Man and Cybernetics Part B-Cybernetics**, Vol 35(6), 1179–1192, 2005.
23. Cai, X., T. Kloks, C.K. Wong, "Time-Varying Shortest Path Problems with Constraints", **Networks**, Vol 29, No 3, 141-150, 1997.
24. Orda, A., R. Rom., "Minimum Weight Paths in Time-Dependent Networks", **Networks**, Vol 21, 295–319, 1991.
25. Halpern, J., "Shortest Route with Time Dependent Length of Edges and Limited Delay Possibilities in Nodes", **Mathematical Methods of Operations Research**, Vol 21, No 3, 117–124, 1977.
26. Cooke, K.L., Halsey E., "The Shortest Route Through a Network with Time-Dependent Internodal Transit Times", **Journal of Mathematical Analysis and Applications**, Vol 14, No 3, 493–498, 1966.