AKILLI TRAFİK KONTROL SİSTEMİ

INTELLIGENT TRAFFIC CONTROL SYSTEM

ZEKİ GÜLER

Yrd. Doç. Dr. Mehmet Demirer

Tez Danışmanı

Hacettepe Üniversitesi

Lisansüstü Eğitim – Öğretim ve Sınav Yönetmeliğinin

Elektrik ve Elektronik Mühendisliği Anabilim Dalı İçin Öngördüğü

YÜKSEK LİSANS TEZİ

olarak hazırlanmıştır.

2013

ZEKİ GÜLER'in hazırladığı "**Akıllı Trafik Kontrol Sistemi**" adlı bu çalışma aşağıdaki jüri tarafından **ELEKRİK ve ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI'**nda **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Başkan

Prof. Dr. Selçuk Geçim

Danışmanı

Yrd. Doç Dr. Mehmet Demirer

Üye

Prof. Dr. Abdullah Çavuşoğlu

Üye

Doç. Dr. Ali Ziya Alkar

Üye

Yrd. Doç Dr. Derya Altunay

Bu tez Hacettepe Üniversitesi Fen Bilimleri Enstitüsü tarafından **YÜKSEK LİSANS TEZİ** olarak onaylanmıştır.

Prof. Dr. Fatma SEVİN DÜZ

Fen Bilimleri Enstitüsü Müdürü

ETİK

Hacettepe Üniversitesi Fen Bilimleri Enstitüsü, tez yazım kurallarına uygun olarak hazırladığım bu tez çalışmasında,

- tez içersindeki bütün bilgi ve belgeleri akademik kurallar çerçevesinde elde ettiğimi,
- görsel işitsel ve yazılı tüm bilgi ve sonuçları bilimsel ahlak kurallarına uygun olarak sunduğumu,
- başkalarının eserlerinden yararlanılması durumunda ilgili eserlere bilimsel normlara uygun olarak atıfta bulunduğumu,
- atıfta bulunduğum eserlerin tümünü kaynak olarak gösterdiğimi,
- kullanılan veriler herhangi bir tahrifat yapmadığımı,
- ve bu tezin herhangi bir bölümünü bu üniversiteye veya başka bir üniversitede başka bir tez çalışması olarak sunmadığımı

beyan ederim.

12/06/2013

Zeki Güler

ÖZET

AKILLI TRAFİK KONTROL SİSTEMİ

ZEKİ GÜLER

Yüksek Lisans, Elektrik Elektronik Mühendisliği Bölümü

Tez Danışmanı: Yrd. Doç. Dr. MEHMET DEMİRER

Haziran 2013, 104 sayfa

Bu çalışmada ulaşım ağlarındaki trafik sıkışıklığını önlemek amacıyla, sıkışıklığa neden olan kavşakların uyarlamalı kontrolünü sağlayan ve kural tabanlı çalışan bir algoritma önerilmiştir. Kavşakta bulunan araçların miktarını (her bir yöndeki kuyruk uzunluğunu) ve bu miktardaki artışı gözlemleyen algoritma en iyi ışık sürelerini bir sonraki döngüde kullanacak şekilde ayarlamaktadır. Böylelikle her kavşağın kendi sinyal sürelerini belirleyebilmesi, kavşakta bekleme sürelerini azaltmakta ve kavşağın en iyi performansta çalışmasını sağlamaktadır.

Kural tabanlı çalışan kavşak modeli oluşturulurken gerçek trafik verilerine benzer veriler kullanılmış ve bu veriler ile eğitilen kural blokları sıradan bir kavşağa göre %20-%30 daha fazla araçın kavşaktan ayrılmasına olanak sağlayan bir algoritma meydana getirilmiştir.

Anahtar Kelimeler: genetik algoritma, bulanık mantık, kavşak kontrolcüsü, akıllı kavaşak yönetimi, uyarlamalı kavşak kontrol, trafik sinyal optimizasyonu

ABSTRACT

INTELLIGENT TRAFFIC CONTROL SYSTEM

ZEKİ GÜLER

Master of Science, Departmet of Electrical and Electrical

Supervisor:Asst. Prof. MEHMET DEMİRER

June 2013, 104 pages

In this study, algorithm is suggested, which works rule based and controls intersections cause traffic, to prevent that traffic in transportation networks. It sets number of cars in intersection (length of line in both sides) and observes the increase of that amount to drive the best period of traffic lights in the next loop. Thus, every intersection's designation of the period of traffic lights decreases the standby time and provides the best performance for the intersection.

Similar data to actual traffic data is used to compose rule based intersection model and rule blocks which are educated with these data generates algorithm that provides 20 to 30 % more cars exit the intersection rather than ordinary intersection.

Keywords: genetic algorithm, fuzzy logic, intersection controller, intelligent traffic controller, adaptive traffic control, traffic optimization

TEŞEKKÜR

Değerli hocam Yrd. Doç. Dr. Mehmet Dermirer ve çalışmamda büyük emeği geçen Prof. Dr. Kemal Leblebicioğlu'na, maddi, manevi her tür desteği sağlayan ORTANA Şirket Müdürü Umut AYDIN'a, ArGe müdürü İbrahim DEMİR'e ve diğer tüm ORTANA ailesine yapmış olduğu yardım ve desteklerinden dolayı teşekkür ederim.

İÇİNDEKİLER

Sayfa

KABUL VE ONAY SAYFASI i

ETİK ii

ÖZET iii

ABSTRACT iv

TEŞEKKÜR v

İÇİNDEKİLER vi

1. GİRİŞ ..... - 1 -

2. YÖNTEM 3

3. TRAFİK SİNYAL KONTROLÜ ÇALIŞMALARINDA KULLANILAN TEMEL PARAMETRELER 6

3.1. Sinyal Döngüsü 6

3.2. Sinyal Döngüsü Zamanı (c) 6

3.3. Trafik Akış Oranı (q_i) 6

3.4. Sinyal Fazı 7

3.5. Doygunluk Debisi (s_i) 8

3.5. Benzeştirici (simulatör) 8

4. ÇÖZÜM İÇİN GELİŞTİRİLEN YÖNTEM VE MODEL 9

4.1. Kayıp Zaman (L) 9

4.2. Etkin Yeşil Süresi (g_e) 9

4.3. Döngü Sonunda Her Bir Yönde Biriken Kuyruk Miktarı (x_i) 9

4.4. Kural Tabanının Oluşturulması 11

4.5. Üyelik Fonksiyonun Belirlenmesi 13

4.6. Bir Adet Kuralın Girdiye Göre Ürettiği Kararın Gösterimi 15

4.7. Birden Fazla Kuralın Durulaştırma İşlemi 19

5. KURALLARIN EĞİTİLMESİ 21

5.1. Çözüm Uzayı 22

5.2. Karmaşık Çok Boyutlu Arama Uzayı 24
5.3. Uygunluk Hesabı 24
5.4. Kromozomlar Arası Genetik Değişim (Çaprazlama, Crossover) 26
5.5. Performans Hesabı 29
5.6. Kaynaştırma İşlemi 30
5.7. Algoritma 31
5.8. Algoritmanın Akış Şeması 31
6. KAYNAŞTIRMA İŞLEMİ VE SONUÇLARI 32
6.1. Eğitim Verisi 32
6.2. Kaynaştırma İşleminin Verime Etkisi 33
6.2.1. Birinci Kaynaştırma İşlemi 37
6.2.2. İki ve Daha Fazla Kaynaştırma İşlemi 39
6.3. Eğitilen Kural Bloklarının Test Verilerine Uygulanması 43
7. SONUÇLAR 44
7.1. Test Verileri İle Yapılan Çalışmaların Sonuçları 44
7.1.1. Birinci Deneme Sonuçları ve Açıklamalar 44
7.1.2. Farklı Test Verileri İle Yapılan Deney Sonuçları 46
7.2. Öneriler ve Tartışma 49
KAYNAKLAR 51
EK - 1 Algoritmanın Yazınsal Gösterimi 53
EK – 2 Algoritmanın Kaynak Kodu 56
ÖZGEÇMİŞ 94

1. GİRİŞ

Hızla gelişen teknoloji, ekonomi ve artan nüfus günümüz şehirlerinde daha fazla ulaşım araçının kullanılmasına neden olmaktadır. Daha fazla araç kullanımı ise kavşaklarda uzun kuyrukların oluşmasına, gereksiz zaman kayıplarına, kavşaklarda beklerken harcanan gereksiz yakıt tüketimine ve çıkan egzoz dumanının doğayı kirletmesine neden olmaktadır [1]. Trafikte az beklemek ve ulaşımı hızlandırmak için daha yeni ve daha geniş yollar inşaa edilebilir fakat günümüz şehirlerinde bunu yapmak hem kolay olmamakta hem de pahalı bir çözüm olarak durmaktadır.

Bilindiği gibi şehir trafiğindeki sıkışıklığın temel nedenini kavşaklar oluşturmaktadır. Bir çok yolun birleştiği noktalar olan kavşaklar ister istemez trafiğin buralarda yavaşlamasına, hatta durmasına sebep olmaktadır. Kavşaklarda bulunan ışıkların yanma sürelerinin sabit ve sıralı olması değişken trafik yoğunluğu ile baş edememekte ve kavşaklarda gereksiz, uzun kuyrukların oluşmasına neden olmaktadır [2].

Trafiğin daha etkin kullanılmasına yönelik ilk çalışmalar 1960'lı yıllarda A.B.D ve Japonya'da başlamıştır. Bu amaca yönelik Zeki Taşıt/Yol Sistemleri ve Özdevinimli Yol Sistemleri gibi kavramlar ortaya atılmıştır [3].

Uyarlamalı trafik kontrol yöntemi, bir başka değişle akıllı trafik kontrol sistemi son yirmi yıldır gelişen teknoloji ile birlikte uygulama alanı bulabilmiştir [4]. Tek bir kavşakta bekleme sürelerini en aza indirmeye çalışan bu tür sistemler, kavşağın yönlerinde araç olup olmamasına bakarak daha önceden belirlenen en az veya en fazla ışık sürelerini uygulamaktadır. Kavşağın en çok araç olan yönüne geçiş hakkı tanıyarak trafik sıkışıklığını önlemeye çalışmaktadır. Bu tür sistemler kavşağın tüm yönlerinde araç varsa her bir yöne en az veya en fazla ışık süresini atayarak sabit zamanlı bir kavşak gibi çalışmaktadır. Bu durum günümüz şehirlerinde her zaman karşılaşılan bir durum olduğu için trafik sıkışıklığına bir çözüm getirmemektedir.

Bir kentin tümünde veya belli bir bölgesinde trafik ışıklarının yanma sürelerine ve birbirlerine göre sinyal sürelerinin kayma miktarına karar verebilmek için öncelikle her kavşağın kendi başına yaptığı davranışlara karar vermek gerekir.

Her kavşağın kendi sinyal sürelerini belirleyebilmesi, araçların kavşakta bekleme sürelerini en aza indirecek ve kavşağın kullanımını, bir başka değişle kavşaktan

ayrılan araç sayısını, en üst seviyeye çıkartacaktır. Bu bağlamda kavşakta oluşabilecek her bir durum için (kavşağın her bir yönündeki araç miktarı, araç miktarındaki artış vs.) ışık sürelerinde bir miktar oynama yapılması gerekmektedir. Kavşakta oluşacak her bir durum bir kural olarak düşünülmüş [5] ve bu kurallar belirli işlemlerden geçirildikten sonra bir karar oluşturulup ışık süreleri elde edilen karara göre yeniden ayarlanmıştır.

Bu çalışmada önerilen çözüm, her kavşak için daha önceden belirlenmiş kurallar ve o kuralın vereceği en uygun kararların olduğu bir kural bloğunun yaratılması ve elde edilen kararlara göre her bir fazdaki ışık sürelerini yeniden ayarlayan bir kontrol algoritmasının uygulanmasıdır. Bir kavşağın her bir yönündeki araç kuyruğu uzunluğu ve kuyruk uzunluğunun artış hızı biliniyorsa bir sonraki döngüde ortalama araç yoğunluğunu en aza indirecek şekilde ışık sürelerini ayarlamak mümkün olmaktadır.

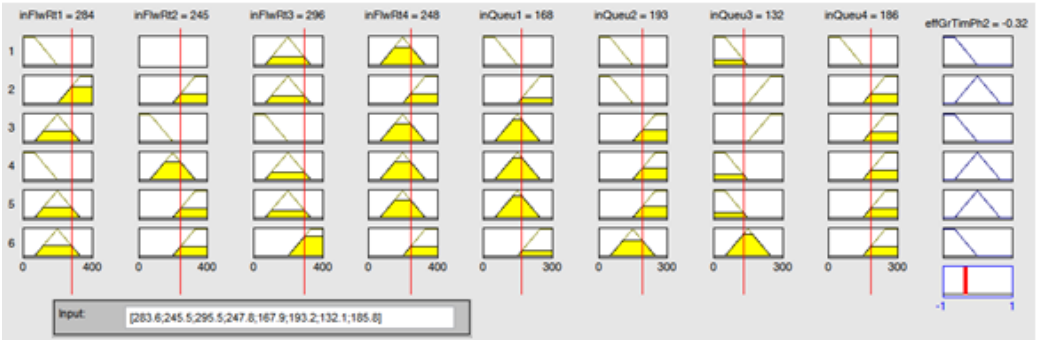
Bu tez çalışması yedi bölümden oluşmaktadır: Birinci bölümde giriş yapılarak problemin tespiti yapılmıştır. İkinci bölümde çözüm için geliştirlen yöntem ve kullanılan araçlar tanıtılmıştır. Üçüncü bölümde kavşak optimizasyon çalışmalarında kullanılan terimler tanıtılmış ve bu çalışmada kullanılan bazı yeni tanımlar eklenmiştir. Dördüncü bölüm ve devamında gelen iki bölüm boyunca geliştirilen çözüm metodunun detaylı analizi yapılmış olup, son bölüm olan yedinci bölümde değişik test verilerileri kullanılarak algoritmanın başarısı test edilmiştir.

2. YÖNTEM

Işık sürelerinin değişen trafik yoğunluğuna uygun bir şekilde ayarlanabilmesi için çok katmanlı bir kural tabanı geliştirilmiştir. Tek bir katmandan oluşan kural yapısı birden fazla kuralın bir araya getirilerek oluşturulduğu bir yapıdır. Her bir kural kavşakta olası bir durumu temsil etmektedir. Kuralların karar kısımları ise kavşağın ışık sürelerinin ilgili durumda ne kadar değişmesi gerektiğine karar vermektedir.

Bir araya gelen kuralların verdikleri kararlar bulanık mantık (fuzzy logic) yapısında ele alınmış ve çıktı olarak tek bir karar elde edilmiştir. Ortaya çıkan bu karar o katmanın oluşturduğu son karardır. Bulanık mantığın temeli bulanık küme ve bulanık alt kümelere dayanır. Kesin ve net sonuçlar yerine yaklaşık sonuçlar elde eder [6]. Bu da tasarlamış olduğumuz sistemin anlık ve belirsiz durumlarda performansını en iyi seviyede tutmaktadır.

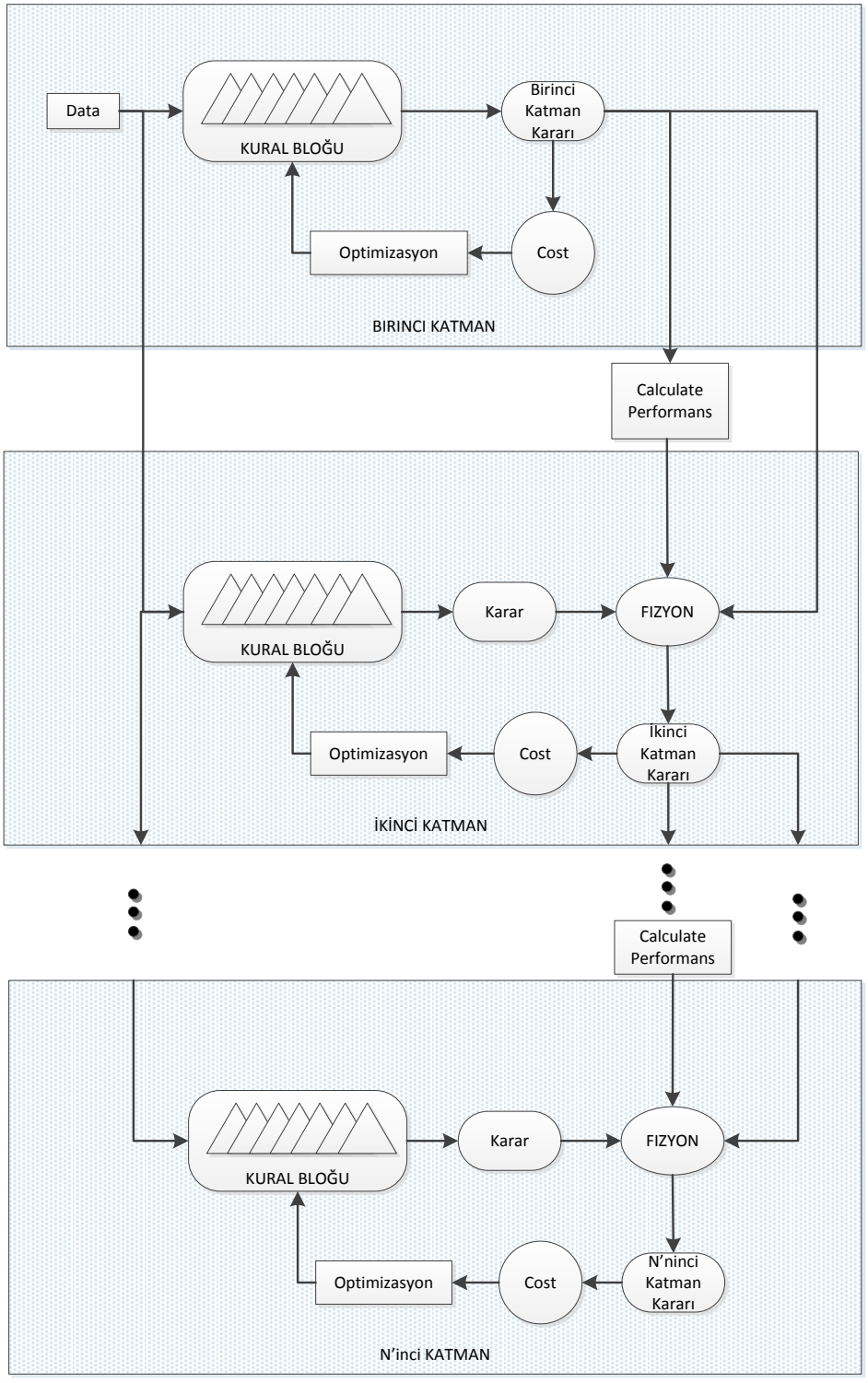
Şekil-1.1'de altı adet kuralın oluşturmuş olduğu bir kural katmanı gözükmektedir. Bir araya gelen kuralların oluşturmuş olduğu katmana blok denmektedir [7]. Bloğun üretmiş olduğu çıktı ise katmanın vermiş olduğu en son karardır.



Şekil-1.1 : Altı Adet kuralın bir araya getirilerek oluşturulduğu kural katmanı örneği

Kuralların vereceği kararlar kavşağın performansının belirlenmesinde kilit rol oynamaktadır. Çıkan kararlara göre kavşağın ışık süreleri artırılır, azaltılılır ya da hiç değişmeden bir sonraki döngüde aynen kullanılır.

Kavşakta gerçekleşebilecek her bir duruma en uygun kararı bulmak zor bir iştir. Bunun yerine olası tüm durumlarda en mantıklı kararları veren kurallar seçilebilir. Nihayetinde bir çok kural karar olarak aynı sonucu üretmektedir. Aynı kararı veren kuralların kural katmanında ikinci kez bulundurulmasına gerek yoktur.



Şekil-2.1 : Uyarlamalı çalışan kavşağın akış şeması

En uygun kuralların seçilebilmesi için optimizasyon teknikleri arasından karmaşık çok boyutlu arama uzayında en iyinin hayatta kalması ilkesine göre bütünsel en iyi çözümü arayan genetik algoritma (GA) kullanılmıştır [9] [10] [11].

Belirli bir sayıda bir araya gelen kuralların oluşturduğu kural katmanı, eğitim verileri ile denenmiş ve başarım en iyi olana kadar katmanın içinde bulunan kurallar bir düzen içerisinde değiştirilmiştir.

Tek bir katmanın oluşturduğu kurallar kavşağın iyi bir verimde çalışması için yeterli değildir. En iyi verimin alınabilmesi için ilk oluşturulan katmanın kuralları kullanıcı tarafından tasarlandıktan sonra, başka diğer katmanların oluşturulması gereklidir. Oluşturulan diğer katmanların kararları bir önceki katmanın kararı ile birleştirilerek kavşağın verimi artırılır.

Her bir katmanda alınan karar bir sonraki katmanın çıktısı ile birleştirilirken bir kaynaştırma (fusion) işleminden geçirilmesi gerekmektedir. Aksi durumda her iki katmanın çıktısı bağımsız ve kendine özgü olmaktadır. Kaynaşma işlemi yapılırken kavşağın ilgili döngü zamanındaki verimi göz önünde bulundurulur. Verimin kötü olduğu yerlerde bir önceki katmanın verdiği karar, iyi olduğu yerlerde ise yeni katmanın verdiği karar ağırlıklandırılarak en son karar üretilir [12].

Kavşağın veriminin bulunabilmesi için bir denetleme kümesine ihtiyaç duyulmuştur. Bu çalışmada denetleme kümesi olarak sıradan bir kavşak seçilmiştir. Sıradan kavşağın özelliği sinyallerinin her zaman sıralı ve sabit süreli olmasıdır. Tüm şartların aynı olduğu durumda, (aynı yönde ve aynı miktarda araç yoğunluğu ve araç kuyruğu uzunluğu olduğu durumda) bir döngü zamanı boyunca, denetleme kümesinde bulunan kavşaktan ayrılan araç miktarının, uyarlamalı yapıda çalışan kavşaktan ayrılan araç miktarına oranı verim ile ilgili bilgiyi vemektedir.

Bu çalışma boyunca amaç verimi uyarlamalı çalışan kavşakta en iyi olacak şekilde artırma çabası olmuştur. Şekil-2.1'de kural katmanlarının oluşturulması, eğitilmesi ve kaynaştırma işlemlerinin akış şeması gözükmektedir.

3. TRAFİK SİNYAL KONTROLÜ ÇALIŞMALARINDA KULLANILAN TEMEL PARAMETRELER

Trafik sinyal optimizasyonu çalışmalarında kullanılan belirli tanım ve ifadeler aşağıda anlatılmıştır. Bu tanım ve ifadelerden çalışma boyunca yararlanılmıştır.

3.1. Sinyal Döngüsü

Bir kavşakta bulanan sinyallerin kendini tekrar ederek yanmaya başlamadan önceki durumuna sinyal döngüsü denir. Her kavşak bünyesinde bulundurduğu sinyalleri belirli bir zaman sonunda yakmalıdır. Böylelikle kavşakta tanımlanmış olan her bir yöne geçiş hakkı tanınmış olacaktır.

Sıradan kavşaklarda sinyallerin yanması belirli bir sıraya göre gerçekleşmektedir. Sinyal gruplarının belirli bir sırada yanması gereksiz yönlere geçiş hakkının tanınmasına neden olabileceği gibi geçiş hakkı tanınması gereken yerlerde gereksiz beklemelere neden olabilir. Bu durum kavşakta uzun kuyrukların oluşmasının ve trafik sıkışıklığının temel nedenini oluşturur.

3.2. Sinyal Döngüsü Zamanı (c)

Kavşaktaki sinyallerin bir sinyal döngüsünü tamamlaması için geçen süreye sinyal döngüsü zamanı denir. Bir sinyal döngüsü zamanında kavşakta bulunan tüm yönlere geçiş hakkı en az bir kere tanınır. Sıradan bir kavşakta bu süre sabit iken uyarlamalı çalışan bir kavşakta bu değer, kavşakta bulunan araç sayısına, kavşağa gelen araç miktarına, kavşağın doygunluk miktarına, eğer evrensel bir çözüm sağlıyorsa diğer kavşakların durumuna ve merkezi bir üniten alınan bilgilere göre değişebilir. Sinyal döngüsü zamanının sabit olduğu uyarlamalı çalışan kavşaklarda mevcuttur [13].

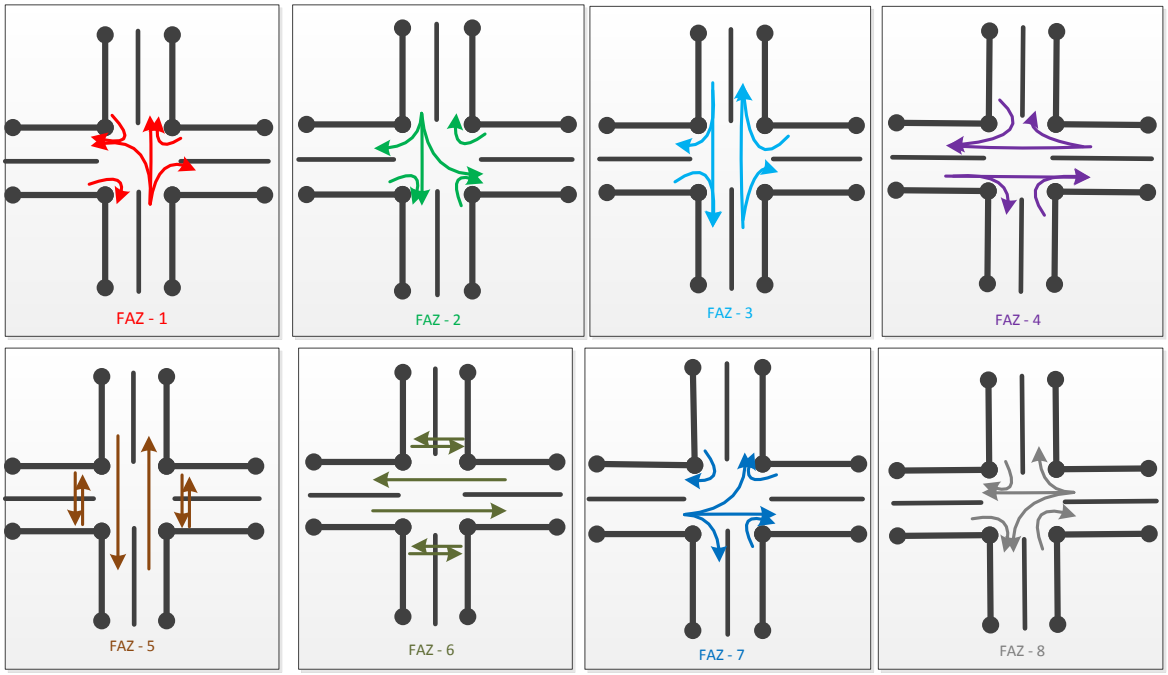
3.3. Trafik Akış Oranı (q_i)

Birim zamanda her bir yönden kavşağa gelen araç miktarına trafik akış oranı denir. Uyarlamalı çalışan kavşaklarda trafik akış oranı sistemin performansının belirlenmesinde kilit rol oynamaktadır [14]. Daha detaylı analizler yapılabilmesi için trafik akış oranındaki artış ya da azalışlar da dikkate alınarak yeni çözüm yolları geliştirilebilir.

3.4. Sinyal Fazı

Bir sinyal döngüsü zamanında aynı renkte yanan sinyallere sinyal fazı denir [1]. Bir kavşağa her bir yönden gelen araçlara aynı anda geçiş hakkının verilmesi kavşaklarda kazaların oluşmasına neden olabilir. Bu yüzden kavşaklarda her bir yöndeki araçlara sırası ile yol verilmelidir. Bir biri ile çakışmayacak şekilde yol verilmesi için kavşaklarda bulunan sinyaller eşzamanlı bir şekilde yanmalıdır. Bir yöne yol verildiğinde çakışmaya neden olabilecek diğer yönlere yol verilmemelidir.

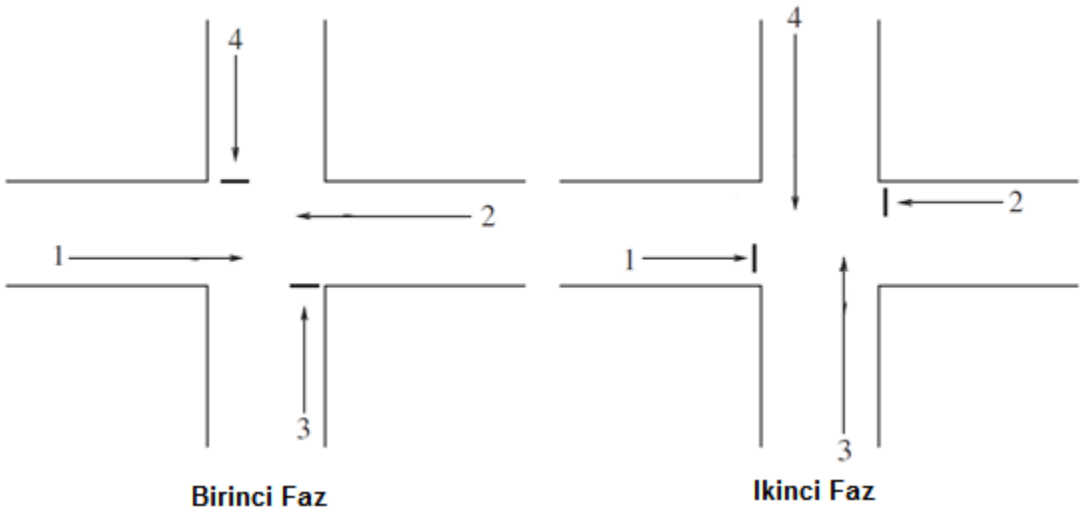
Sıradan bir kavşak modelinde Şekil-3.1'de gösterildiği gibi 8 adet faz bulunabilir. Daha fazla sayıda faz yaratmak mümkündür fakat bu durum anlamlı olmamaktadır.



Şekil-3.1 : Bir kavşakta bulunabilecek faz adedi ve yönleri

Şekil-3.1 incelendiğinde görülecektir ki temelde bir kavşak iki fazdan oluşmaktadır. Dördüncü faz, altıncı faz, yedinci faz ve sekizinci faz doğu ve batı yönüne araç geçişlerine izin verirken, birinci faz, ikinci faz, üçüncü faz ve beşinci faz ise kuzey ve güney yönüne araç geçişlerine izin verir. Bu durumda bir kavşağın temel olarak iki fazdan oluştuğu kabul edilebilir. İki fazdan alınacak verim en iyi olarak ayarlandıktan sonra alt fazların süreleri hesaplanabilir.

Şekil-3.2'de iki adet fazdan oluşan bir kavşak modeli gösterilmektedir. Birinci faz devresinde araçların bir ve iki yönüne geçişlerine izin veriliyorken; ikinci faz aşamasında araçların üç ve dört yönüne geçişlerine izin verilmektedir.



Şekil-3.2 : İki fazlı kavşak modeli

3.5. Doygunluk Debisi (s_i)

Bir kavşaktan belli bir yöne birim zamanda çıkabilecek azami araç sayısı belirlidir. Doygunluk debisi olarak adlandırılan bu oran, kavşağın fiziki yapısına bağlıdır [15]. Her bir kavşak kendine özgü bir fiziki yapıya sahip olduğu için kavşakların doygunluk debileri farklılık gösterecektir. Bu durum her kavşak için yeniden en iyileme hesaplamalarının yapılması gerektiğini gösterir.

Kavşağa birim zamanda doygunluk debisinden daha az miktarda araç geliyorsa ve ilgili yönde herhangi bir araç bulunmuyorsa kavşaktan ayrılan araç miktarı ilgili yöne yeşil yandığı sürece trafik akış oranına eşit olacaktır. Birim zamanda doygunluk debisinden daha fazla miktarda araç geliyorsa, kavşaktan ayrılan araç miktarı doygunluk debisine eşitlenecek ve kavşakta kuyruklanmalar oluşacaktır.

3.5. Benzeştirici (simulatör)

Akıllı trafik denetcisi tasarımında yapılan çalışmaları test edebilmek ve akıllı kavşağın sıradan bir kavşağa göre başarımını gözlemleyebilmek için bir trafik simulatörü geliştirilmiştir. Geliştirilen bu simulatör bir kavşağın temel fonksiyonlarını yerine getirmekte ve geliştirilen algoritmaların sonuçları gözlemlenebilmektedir.

4. ÇÖZÜM İÇİN GELİŞTİRİLEN YÖNTEM VE MODEL

Daha öncede belirtildiği gibi bir kavşak iki, üç, dört ya da daha çok fazlı olacak şekilde modellenebilir. Bir sinyal döngüsü birden fazla faz oluşturabilecek şekilde bölünebilir. Bu çalışmada kullanılan modelde bir döngü süreci iki fazlı olacak şekilde seçilmiştir. Birinci faz boyunca bir ve iki nolu yönlere geçiş hakkı tanınırken, ikinci faz boyunca üç ve dört nolu yönlerine geçiş hakkı tanınmıştır (Şekil-3.2).

4.1. Kayıp Zaman (L)

Bir sinyal döngüsü zamanında etkin olarak kullanılmayan zamana kayıp zaman denir. Örneğin bir yöne geçiş hakkı verildiğinde kavşakta bulunan araçlar hemen harekete geçemedikleri için geçiş hakkı boyunca bir zaman kaybı oluşacaktır [16].

4.2. Etkin Yeşil Süresi (g_e)

Bir sinyal döngüsü boyunca etkili olan yeşil süresine etkin yeşil süresi denir. Etkin yeşil süresi aşağıdaki şekilde hesaplanır [17].

$$g_{ei} = g_i + y_i - L \tag{1}$$

i : faz indisi, y : faz boyunca yanacak yeşil süresi, L : kayıp zaman

4.3. Döngü Sonunda Her Bir Yönde Biriken Kuyruk Miktarı (x_i)

Kavşakta bir döngü zamanı sonunda her bir yönde biriken kuyruk miktarı aşağıdaki şekilde hesaplanmıştır.

$$x_1(k+1) = x_1(k) + q_1 g_{e_2(k)} + q_1\left(c - g_{e_2(k)}\right) - s_1\left(c - g_{e_2(k)}\right)$$
$$x_2(k+1) = x_2(k) + q_2 g_{e_2(k)} + q_2\left(c - g_{e_2(k)}\right) - s_2(c - g_{e_2(k)})$$
$$x_3(k+1) = x_3(k) + q_3 g_{e_2(k)} + q_3\left(c - g_{e_2(k)}\right) - s_3 g_{e_2(k)}$$
$$x_4(k+1) = x_4(k) + q_4 g_{e_2(k)} + q_4\left(c - g_{e_2(k)}\right) - s_4 g_{e_2(k)}$$

c: döngü zamanı, g_{ei}: etkin yeşil süresi, x_i: bir döngü sonunda ilgili yönde biriken kuyruk miktarı, s_i: ilgili yönün doygunluk debisi, q_i : trafik akış oranı .

Gerekli düzenlemeler yapıldıktan sonra sistemin durum denklemi aşağıdaki gibi olmaktadır.

$$\bar{x}(k+1) = A(k)\overline{x}(k) + B\bar{u}(k) \quad (2)$$

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, B = \begin{bmatrix} s_1 & q_1 - s_1 \\ s_2 & q_2 - s_2 \\ -s_3 & q_3 \\ -s_4 & q_4 \end{bmatrix}, \bar{u}(k) = \begin{bmatrix} g_{e_2(k)} \\ c \end{bmatrix},$$

Çizelge 4.1 : Kural bölütü

Öncü Etiketi	Yoğunluk			Açıklama
inFlwRt1	az	orta	çok	Bir yönündeki araç yoğunluğu
inFlwRt2	az	orta	çok	İki yönündeki araç yoğunluğu
inFlwRt3	az	orta	çok	Üç yönündeki araç yoğunluğu
inFlwRt4	az	orta	çok	Dört yönündeki araç yoğunluğu
queLine1	az	orta	çok	Bir yönünde biriken araç yoğunluğu
queLine2	az	orta	çok	İki yönünde biriken araç yoğunluğu
queLine3	az	orta	çok	Üç yönünde biriken araç yoğunluğu
queLine4	az	orta	çok	Dört yönünde biriken araç yoğunluğu

Çizelge-4.2 Kural bölütünün verdiği karar

Değişim	Sonuç			Açıklama
ΔeffGrPh2	Azalt	Değiştirme	Artır	ikinci fazın etkin yeşil süresinin ne kadar değiştirileceği. Birinci fazın etkin yeşil süresi döngü zamanından çıkartılarak kolayca bulunabilir.

4.4. Kural Tabanının Oluşturulması

Bir kavşağın olası tüm girdilerine karşı en uygun etkin yeşil süreleri bulunursa o kavşağın en iyi verimde çalışması sağlanabilir. Bir başka değişle bir döngü içerisinde gelen araç yoğunluğuna ve kuyrukta bekleyen araç miktarına göre etkin yeşil süresi değiştirilebilirse kavşak istenen verimde çalışacaktır. Bü yüzden olası tüm durumlar çıkartılmalı, her durum için etkin yeşil süresi daha önceden hesaplanıp bir yerde tutulmalıdır fakat olası her durum için en uygun kararın bulunması, kararların ve durumların bir hafıza modülünde saklanması az bir kaynak gücüne sahip kavşak denetçileri ile mümkün olmamaktadır. Nihayetinde kavşaklarda çalışan cihazlar belirli bir işlemci gücüne ve hafıza yerine sahiptir. Daha güçlü işlemciler ve büyük boyutlu bellek modülleri kullanılabilir fakat bu da beraberinde ek maliyete ve hata yapabilme ihtimalinin artmasına neden olacaktır.

Bir kavşağın her hangi bir yönünden gelen araç yoğunluğu ve kuyruklanma miktarı AZ, ORTA ve ÇOK olacak şekilde sınıflandırılabilir. Belirtilen sınıflandırma çözüm için geliştirilen yöntemin kural tabanının bir bölütünü oluşturmaktadır. Bu bölütün tüm yönlerdeki araç yoğunluğu ve kuyruklanma miktarı Çizelge-4.1'de gösterildiği gibidir. Çizelge-4.1' de belirtilen kural bölütünün oluşturacağı karar ise Çizelge-4.2'de gösterilmiştir.

Ele alınan kavşak iki fazlı bir model olduğu için ikinci fazın etkin yeşil süresi bulunduktan sonra birinci fazın etkin yeşil süresi aşağıdaki gibi hesaplanır.

$$effGrPh1 = c - (effGrPh2 + \Delta effGrPh2) \quad (3)$$

c : döngü süresi, effGrPh1: birinci fazın etkin yeşil süresi, effGrPh2: ikinci fazın etkin yeşil süresi, ΔeffGrPh2 : ikinci fazın etkin yeşil süresi için kuralların verdiği karara göre bulumuş değişim miktarı

Elde edilen veriler ışığında her bir girdi için bir kural oluşturulmuştur. Rasgele seçilmiş kuralların oluşturduğu kural bölütü (4) nolu denklemde gösterildiği gibidir.

Kural içerisinde dokuz adet değişken bulunmaktadır. q_1, q_2, q_3, q_4, x_1, x_2, x_3, x_4, öncül değişkenleri gösterirken, ΔeffGrPh2 ise karar değişkenini gösterir.

Oluşturulan kural tabanı bir girdi ile beslendiğinde bir karar üretir. **THEN** operatörü olarak adlandırılan bu işlem öncül değişkenlerin bir kaç işlemden geçirilmesi ile üretildiğini gösterir. Bu işlem basitçe asgari değer üreten üyelik fonksiyonun kararı

belirlediği bir seçim işlemidir. Öncül değişkenlerinin, üyelik fonksiyonu (membership) derecelerinin en küçük olanının seçildiği işlemler ise **AND** operatörü ile gösterilmiştir.

IF

$$(q_1 = i1\ \boldsymbol{AND}\ q_2 = i2\ \boldsymbol{AND}\ q_3 = i3\ \boldsymbol{AND}\ q_4 = i4\ \boldsymbol{AND}\ x_1 = i5\ \boldsymbol{AND}\ x_2 = i6\ \boldsymbol{AND}\ x_3 = i7\ \boldsymbol{AND}\ x_4 = i8)$$

THEN

$$(\Delta effGrPh2 = res)$$

(4)

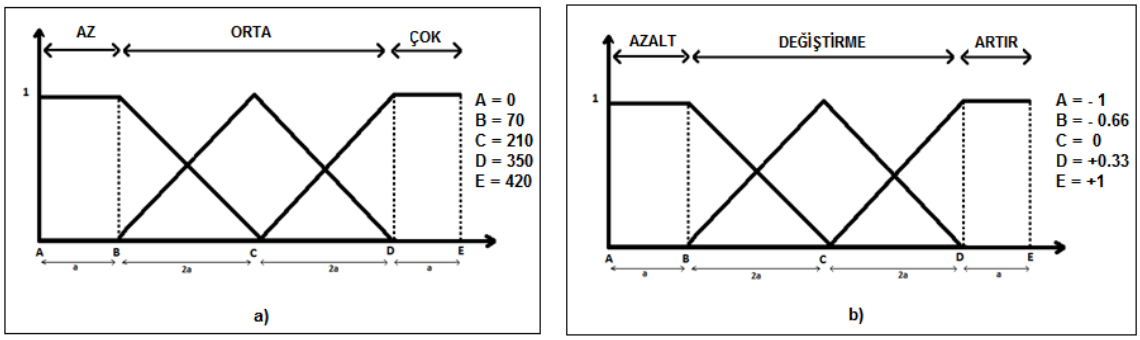
Doğu ve batı yönünde, bir başka bir değişle bir nolu fazın geçiş hakkı tanıdığı yönlerde (Şekil 3-2), araç trafiğinin yoğun ve bu yönde bekleyen araç sayısının fazla olan kavşağın, bir sonraki döngüsünde bir ve iki yönündeki etkin yeşil süresini artırması beklenir. Çizelge-4.3'de bu durumu gösteren bir örnek görülmektedir.

Çizelge-4.3 Birinci faz yönünde yoğun trafiğin olduğu durumda kuralların ve kararların oluşturulması örneği

	q_1	q_2	q_3	q_4	x_1	x_2	x_4	x_4	ΔeffGrPh2
KURAL	Çok	Çok	Az	Orta	Çok	Orta	Az	Az	Azalt

4.5. Üyelik Fonksiyonun Belirlenmesi

Kural tabanını girdi ve çıktı üyelik fonksiyonları Şekil-4.1'deki gibi seçilmiştir.



Şekil-4.1 a) Girdi için kullanılan üyelik fonksiyonu grafiği b) Çıktı için kullanılan üyelik fonksiyonu grafiği

Üyelik fonksiyonu seçilirken iki tip fonksiyon tercih edilmiştir: Üçgensel fonksiyon (triangular) ve eşkenar yamuk fonksiyon (trapezoidal). Her bir fonksiyonun altında kalan alan birbirine eşit olacak şekilde girdi değerleri ayarlanmıştır. Böylelikle karmaşık çok boyutlu arama uzayında her bir olası arama için eşit bir ağırlıklandırma yapılmıştır. Bir başka değişle her bir sınıf için gelen girdilere eşit değerlendirme yapılmıştır [18].

Bir önceki örneğe geri dönersek kural tabanında gerçekleştirilen işlem Çizelge-4.4 de gösterildiği gibidir.

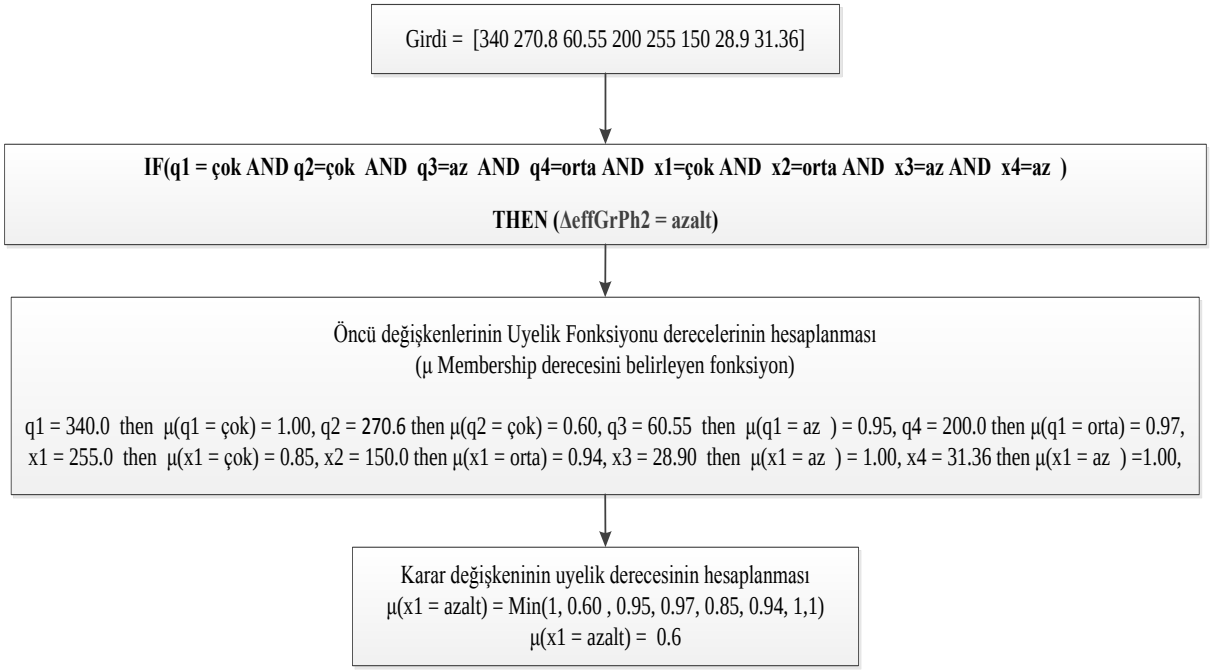
Çizelge-4.4 Kural tabanında gerçekleşen işlemler

inFlwRt1	inFlwRt2	inFlwRt3	inFlwRt4	queLine1	queLine2	queLine3	queLine4	THEN	ΔeffGrPh2
Çok	Çok	Az	Orta	Çok	Orta	Az	Az		Azalt
340	270,8	72	205	350	210.8	20.5	55.8		-0,6

Çizelge-4.4 incelendiğinde bir yönündeki fazda yoğun bir trafiğin mevcut olduğu ve birinci faza ait yönlerde kuyruklanmanın çok olduğu, fakat ikinci faz yönünde

trafik yoğunluğunun az ve kuyruk miktarının düşük olduğu kolayca anlaşılabilir. Bu durumda kuralın vereceği karar ikinci fazın etkin yeşil süresini azaltmak olmalıdır.

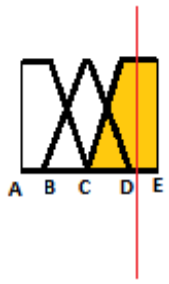
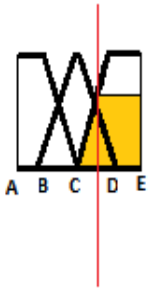
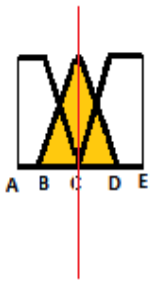
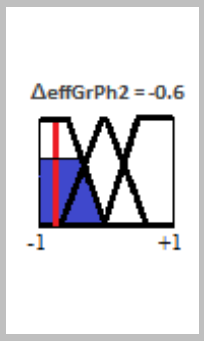
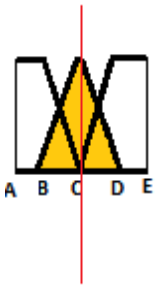
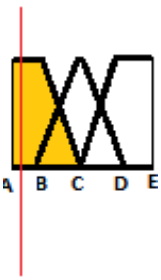
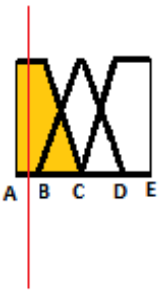
İkinci fazın etkin yeşil süresi azaltıldığında, bir nolu fazın etkin yeşil süresi artırılmış olacak ve trafik Şekil-3.2 de gösterilen bir ve iki yönünde rahatlayacaktır. Yapılan işlemin akış şeması Şekil-4.2 de gösterilmektedir.



Şekil-4.2 Bir adet kuralın üyelik fonksiyonu derecelerinin hesaplanması ve kararların oluşturulması

Yukarıda verilen örnekte bir adet kuralın girdiye göre üretmiş olduğu kararın nasıl hesaplandığı gösterilmiştir. İkicni bölümde anlatıldığı gibi kural katmanı birden fazla kuraldan meydana gelmektedir. Birden fazla kuralın meydana getirdiği katmanın vereceği kararın nasıl hesaplandığı ise bir sonraki bölümde anlatılmıştır. Uygulanan çözüm metodunun daha iyi anlaşılır kılınması için tek bir kuralın üyelik fonksiyonunun girdiye göre ürettiği sonuç ve yöntem Çizelge-4.5 gösterilmiştir.

Çizelge-4.5 Tek bir kuralın girdi ve karar yapısı

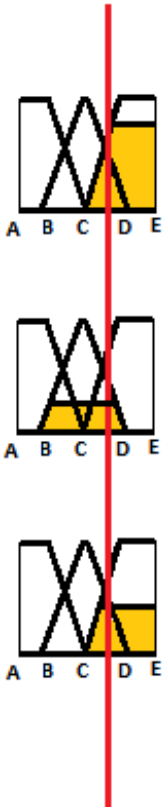
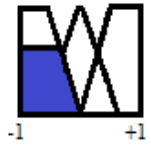
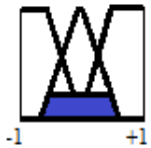
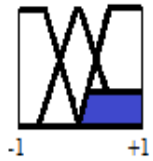
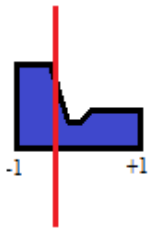
KURAL				KARAR
q_1	q_2	q_3	q_4	**ΔeffGrPh2**
Çok	Çok	Az	Orta	Azalt
340	270.8	72	205	-0,6
				
x_1	x_2	x_3	x_4	
Çok	Orta	Az	Az	
350	210.8	20.55	55.80	
				

.

4.6. Bir Adet Kuralın Girdiye Göre Ürettiği Kararın Gösterimi

Girdiye göre kuralın üretmiş olduğu karar bulunurken üyelik fonksiyonlarından o karar için aktif olan üyelik fonksiyonu bir değer üretmiş ve bu değerlerden en azına sahip olanının karar öncülünde kestiği yer seçilerek (**AND** operasyonu gerçekleştirilerek) o kuralın nihayi kararı olarak atanmıştır [19]. Çizelge-4.5'de bu değer iki yönünde gelen trafik akış yoğunluğu öncül değişkenin vermiş olduğu karar en son kararın belirlenmesinde etkili olmuştur.

Çizelge-4.5’de sarı renkte gösterilen üyelik fonksiyonları o kural için tanımlanmış aktif üyelik fonksiyonlarıdır. Karar değişkeni için üyelik fonksiyonunun bulduğu değer, belirtilen durum için alınacak kararın hesaplanmasında kullanılacaktır. Etkin yeşil süresinin bulunabilmesi için bir veya birden fazla kuralın verdiği kararların durulaştırma (defusification) işleminden geçirilmesi gerekmektedir [20]. Bir sonraki bölümde birden fazla kuralın durulaştırma işlemi ve kural katmanının en son sonuca nasıl ulaştığı anlatılmaktadır.

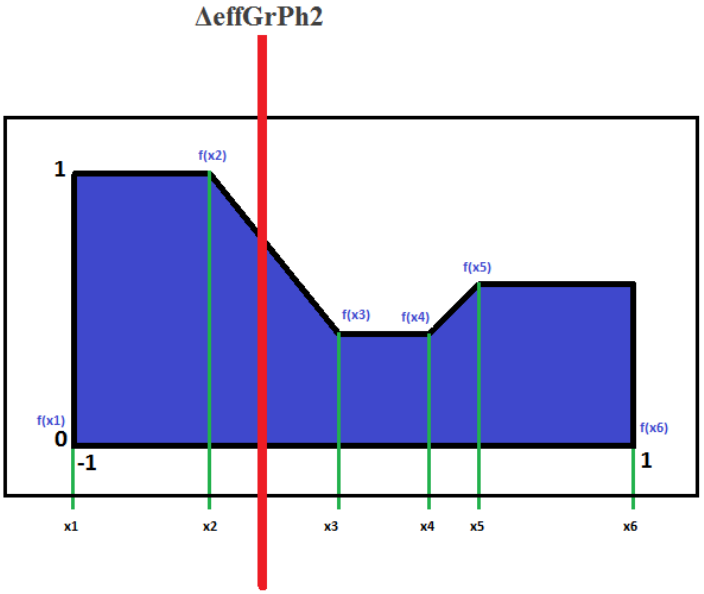
q1	q2	q3	q4	x1	x2	x3	x4	effGrPh2
								
								
								
• Karar kısımlarının birleştirilmiş hali. Bir başka değişle kuralların verdiği kararların toplamı. • Karar kısımları birleştirilirken en fazla alan metodu kullanılmıştır. • Durulaştırma işlemi uygulandıktan sonra kural bloğunun verdiği nihayi karar bulunur. • Nihayi karar yandaki şekilde grafiği dik kesen çizgi (kırmız cizgi) ile gösterilmektedir.								

Çizelge-4.6 Birden fazla kuralın durulaştırma işlemi

4.7. Birden Fazla Kuralın Durulaştırma İşlemi

Çizelge-4.6'da üç adet kuralın durulaştırma işlemi gözükmektedir. Girdiye göre aktif olan üyelik fonksiyonları ve her bir kuralın vermiş olduğu karar kısımlarının bir örneği Çizelge-4.6'da gösterilmiştir. Birleştirme işlemi yapılırken en büyük metodu (fonksiyonun altında kalan alanlardan en büyük olanı) seçilmiştir. Durulaştıma işlemi yapılırken kullanılan denklem ve grafiksel anlatımı aşağıdaki gibidir.

$$\Delta effGrPh2 = \frac{\sum_i (\int_{-1}^{1} x_i f_i(x_i) dx)}{\sum_i (\int_{-1}^{1} f_i(x_i) dx)} \quad (5)$$



Şekil-4.3 Üç adet kuralın en büyük metoduna göre bulunmuş grafiği

Yukarıda gösterilen grafikte etkin yeşil süresi aşağıdaki şekilde hesaplanır.

$$\Delta effGrPh2 = \frac{x_1 f(x_1) + x_2 f(x_2) + x_3 f(x_3) + x_4 f(x_4) + x_5 f(x_5) + x_6 f(x_6)}{f(x_1) + f(x_2) + f(x_3) + f(x_4) + f(x_5) + f(x_6)}$$

İkinci faz için bir sonraki döngüde uygulanacak etkin yeşil süresi aşağıdaki denklem ile bulunur.

$$newEffGrPh2 = oldEffGrPh2 * (1 + \Delta effGrPh2) \quad (6)$$

veya

$$newEffGrPh2 = oldEffGrPh2 + \Delta effGrPh2 * \alpha \quad (7)$$

Durulaştırma işlemi sonunda -1 ile +1 arasında bir sonuç elde edilmektedir. Çıkan bu sonuca göre etkin yeşil süresindeki değişimin ne kadar olacağı belirlenir. PID

denetçi kullanılacaksa (6) nolu denklem; PI denetçi kullanılacaksa (7) nolu denklem kullanılır. Durulaştırma sonucuna karşılık gelen etkin yeşil süresindeki örnek değişimler Çizelge-4.7'de gösterilmiştir

Çizelge 4.7: Durulaştırma işlemi sonucu ikinci fazdaki etkin yeşil süresindeki değişim örneği

Durulaştırma sonucu bulunan değer	PID	PI
0	değişiklik yapma	değişiklikyapma
-0.25	Bir önceki etkin yeşil süresini %25 azalt	Bir önceki etkin yeşil süresini belirli bir değerin (α) %25'i kadar azalt
0.75	Bir önceki etkin yeşil süresini %75 artır	Bir önceki etkin yeşil süresini belirli bir değerin (α) %75'i kadar artır
1	Bir önceki etkin yeşil süresini %100 artır	Bir önceki etkin yeşil süresini belirli bir değer (α) kadar artır.
-1	Bir önceki etkin yeşil süresini %100 azalt	Bir önceki etkin yeşil süresini belirli bir değer (α) kadar azalt

5. KURALLARIN EĞİTİLMESİ

Bir önceki kısımda anlatıldığı gibi bir kaç durum için karar üretmek basittir, fakat gerçek bir çözüm için tüm koşulların ele alınması ve kuralların karar kısımlarının en uygun şekilde tasarlanması gerekir. Tasarlanmış olan sistem şu ana kadar sadece sekiz adet girdiden ve bir adet çıktıdan oluşan bir sistemdir. Basit düşünülmüş bir sistem için bile en azından 3^8 farklı kombinasyon oluşmaktadır. Her bir kombinasyon çıkartılıp, her bir durum için "en mantıklı" karar tasarlansa bile bu sadece tek bir kavşak için geçerli bir sonuç olacaktır. Çünkü kavşakta oluşan kuyruğun ve kavşaktan ayrılan araç miktarının hesabı yapılırken her bir kavşağın doygunluk debisi (saturation flow rate) farklılık gösterecektir. Bir başka ifadeyle her bir kavşak için en uygun kararları veren kurallar tek tek tasarlanmalıdır.

Bahsedilen işlemlerin çokluğu ve zorluğu dışında bir kavşağın verimini artırmak için sadece trafik yoğunluğuna ve kuyruklanma miktarına bakmak yeterli olmayabilir. Daha iyi bir verim alabilmek için iki döngü zamanı önce gelen araç miktarı ile bir döngü zamanı önce gelen araç miktarındaki artış ya da azalışta sistemin bir girdisi olarak değerlendirilebilir. Böylelikle trafik yoğunluğundaki anlık sıçramaya daha yumuşak bir çevap üretilebilir. Bu durumda işleme dahil olan kural adedi 3^{12} adede çıkmaktadır. Bu kadar miktarda veriyi işlemek, saklamak ve o anki duruma göre en uygun sonucu bulup uygulamak çok fazla işlem ve zaman gerektirmektedir.

Çizelge-5.1 Kural bölütünün kromozomsal gösterimi

	q_1	q_2	q_3	q_4	x_1	x_2	x_3	x_4	**ΔeffGrPh2**
KURAL	çok	Çok	az	orta	çok	orta	az	az	Azalt
KROMOZOM	11	11	00	01	11	01	00	00	00

Kurallar çıkartılmaya ve gözlemlenmeye başlandığında bazı farklı durumların benzer kararlar ile çözümlenebildiği görülmektedir. Bu durumda benzer karar veren kuralları gruplamak ve o anki gerçek zamanlı girdiye en yakın kuralın

kararını uygulamak bir çözüm olabilir. Böylelikle daha az miktarda kural işleme dahil olacak ve sistemin işleyiş hızı artırılacaktır. Daha az kural adedi kavşakta bulunan denetçi için daha az hafıza bölgesi anlamına gelecektir. Bu durumda da her bir olası girdi için "en uygun kararların" verildiği kuralların tasarlanması ve benzer çıktıların gruplandırılması bir işi yükü olarak durmaktadır.

Bir başka çözüm yöntemi ise başlangıçta "işe yarar" belirli bir sayıda kuralı bir araya getirerek bir kural tabanı oluşturmaktır. Karar kısımları gider fonksiyonu değerini en aza indirecek şekilde tasarlanır. Bu şekilde tasarlanmış kurallara "İşe yarar kural" denir. Böylelikle çözüm kümesinde çözüme en yakın noktadan bir başlangıç yapılmış olunur. Çözüme giden bir sonraki adımı atarken bu noktadan daha iyi bir yöne gitmek geliştirilen algoritmanın amacı olacaktır. Başlangıçta seçilen ve işe yarar kuralların oluşturduğu bloğa daha sonra rasgele seçilmiş kurallar kaynaştırılmıştır. Rasgele seçilen kuralların karar kısımları "en doğru" olacak şekilde tasarlanmış ise algoritmanın atacağı bir sonraki adım, çözüme daha yakın bir nokta olacaktır.

Bu nokta şu sorunun cevabı önem kazanmaktadır: "İşe yarar" kural setleri nasıl oluşturulacaktır? Başlangıçta belirli sayıda kural seti her bir kavşak için seçilir. Kurallar seçilirken her bir durum için en iyi kararın verildiği durumlar olduğu göz önünde bulundurularak tasarlanır. Böylelikle çözüm kümesine en yakın noktadan başlangıç yapılmış olunur. Başlangıç için tasarlanan otuz adet kurallın bir örneği Çizelge-5.2'de gösterilmektedir. Başlangıçta seçilen ve *işe yarar* kuralların oluşturduğu bloğa daha sonra rasgele oluşturulmuş kurallar kaynaştırılmıştır. Kaynaştırılmadan önce genetik algoritma yardımı ile kavşağın ortalama kuyruk miktarı, bir başka değişle gider fonksiyonu (Cost) değeri, en az olacak şekilde ve performans değeri gözetilerek optimize edilmiştir.

5.1. Çözüm Uzayı

Daha öncede belirtildiği gibi 3^8 adet kural kümesi için uygun kararların üretilmesi zahmetli bir iştir. Bu tür verilerin işlenmesi için evrensel bir çözüm metodu olan genetik algoritma (GA) kullanılmıştır.

Sistemin girdisi olan araç yoğunluğu ve kuyruk uzunlukları (Az, Orta, Çok) ve sistemin çıktısı olan etkin yeşil süresindeki değişim miktarı (Azalt, Değiştirme, Artır) "iki ikil (bit)" ile temsil edilmiştir. Bu durumda GA'nın bir kromozomu 9

parçadan (Genden) oluşmakta ve her bir parça iki ikil ile temsil edilmektedir. GA'nın kural bölütünün kromozom olarak gösterimi Çizelge-5.1'de gösterilmektedir.

Çizelge-5.2 Baslangic için tasarlanan otuz adet kural

	q1	q2	q3	q4	x1	x2	x3	x4	Karar
Kural 1	AZ	ORT	ORT	ORT	AZ	AZ	AZ	AZ	**ART**
Kural 2	AZ	COK	ORT	ORT	AZ	AZ	AZ	AZ	**DEG**
Kural 3	ORT	ORT	ORT	AZ	AZ	AZ	ORT	AZ	**DEG**
Kural 4	ORT	ORT	ORT	ORT	AZ	AZ	AZ	AZ	**DEG**
Kural 5	AZ	ORT	ORT	ORT	AZ	AZ	AZ	AZ	**ART**
Kural 6	ORT	ORT	ORT	ORT	ORT	ORT	ORT	ORT	**AZL**
Kural 7	AZ	AZ	AZ	COK	AZ	AZ	AZ	AZ	**DEG**
Kural 8	AZ	AZ	ORT	ORT	AZ	AZ	ORT	COK	**DEG**
Kural 9	AZ	AZ	ORT	COK	AZ	AZ	ORT	COK	**DEG**
Kural 10	AZ	AZ	ORT	COK	AZ	AZ	ORT	COK	**DEG**
Kural 11	AZ	AZ	COK	COK	AZ	AZ	ORT	COK	**ART**
Kural 12	AZ	COK	AZ	AZ	AZ	COK	ORT	ORT	**AZL**
Kural 13	AZ	COK	AZ	AZ	ORT	COK	ORT	ORT	**AZL**
Kural 14	AZ	COK	AZ	AZ	ORT	COK	ORT	ORT	**ART**
Kural 15	ORT	COK	AZ	ORT	COK	COK	ORT	ORT	**AZL**
Kural 16	COK	COK	COK	ORT	AZ	ORT	ORT	AZ	**AZL**
Kural 17	COK	ORT	AZ	AZ	ORT	ORT	AZ	AZ	**DEG**
Kural 18	COK	AZ	COK	AZ	ORT	ORT	ORT	ORT	**ART**
Kural 19	ORT	ORT	ORT	AZ	AZ	AZ	ORT	ORT	**AZL**
Kural 20	AZ	COK	ORT	ORT	AZ	AZ	COK	ORT	**AZL**
Kural 21	COK	AZ	ORT	ORT	AZ	COK	COK	ORT	**AZL**
Kural 22	AZ	AZ	AZ	AZ	COK	ORT	AZ	ORT	**DEG**
Kural 23	AZ	AZ	AZ	AZ	COK	ORT	ORT	ORT	**DEG**
Kural 24	COK	AZ	AZ	AZ	COK	COK	COK	COK	**AZL**
Kural 25	COK	AZ	AZ	ORT	COK	COK	COK	COK	**AZL**
Kural 26	COK	COK	AZ	AZ	COK	COK	COK	COK	**AZL**
Kural 27	COK	COK	AZ	AZ	COK	COK	COK	COK	**AZL**
Kural 28	COK	COK	ORT	AZ	COK	COK	COK	COK	**AZL**
Kural 29	COK	COK	COK	AZ	COK	COK	COK	COK	**AZL**
Kural 30	COK	COK	COK	ORT	COK	COK	COK	COK	**DEG**

5.2. Karmaşık Çok Boyutlu Arama Uzayı

Kuralların oluşturduğu havuz, problemin uygun çözümlerinin uzayını meydana getirir. Bu havuzdaki her bir kromozom uygun birer çözümdür. Her bir "uygun çözüm" için bir uygunluk (**fitness)** hesabı yapılır ve ilgili kromozom bu uygunluk değeri ile işaretlenir.

GA'nın burada yerine getirdiği görev iyi sonucu veren kromozomları tutmak ve bunların oluşturacağı yeni nesli bir sonraki döngüde kullanarak en iyi çözümü bulmaktır. Uygunluk hesabı yapılırken kötü sonuç veren kromozomlar arama uzayından atılır.

Geliştirilen algoritmada önerilen çözüm, birden fazla kuralın en iyi sonucu verecek şekilde seçilmesi ve karar kısımlarının ayarlanması olduğu için kromozom tek bir kuralı değil birden fazla kuraldan meydana gelen bir kural bloğunu temsil etmektedir.

5.3. Uygunluk Hesabı

Uygunluk hesabı yapılırken kavşağın N adet sinyal döngüsü zamanı sonundaki toplam kuyruklanma miktarı ele alınmıştır. Bir başka değişle, her döngü sonunda biriken araç miktarı toplanarak bir gider fonksiyon değeri elde edilmiştir. N adet döngü sonucu oluşan gider fonksiyon değeri sistemin uygunluk değerinin bulunmasında kullanılmıştır.

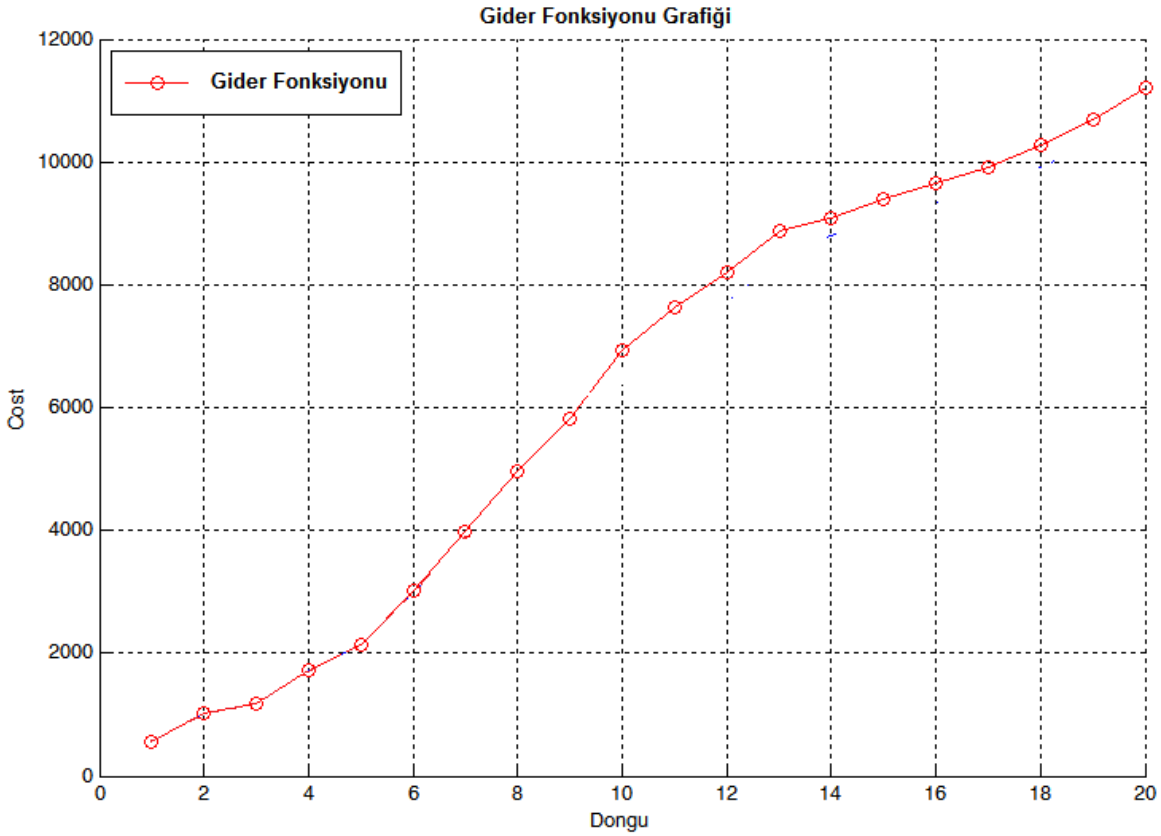
$$uygunluk\ değeri = \frac{1}{gider\ fonksiyon\ değeri} \tag{8}$$

Gider fonksiyonu değeri ne kadar küçük olursa uygunluk değeri o kadar büyük olacaktır. Bu da kavşakta her bir döngü zamanı sonunda beklemelerin az, kuyruklanmanın kısa ve kavşağın performansının iyi olacağı anlamına gelmektedir.

Toplam gider değerini hesaplamak için aşağıdaki formül kullanılmıştır:

$$Toplam\ Gider = C_1 + C_2 + \cdots + C_N \quad (9)$$

Şekil-5.1'de bir kavşağın yirmi adet döngü sonunda toplam kuyruk miktarındaki artışı gösterilmektedir.

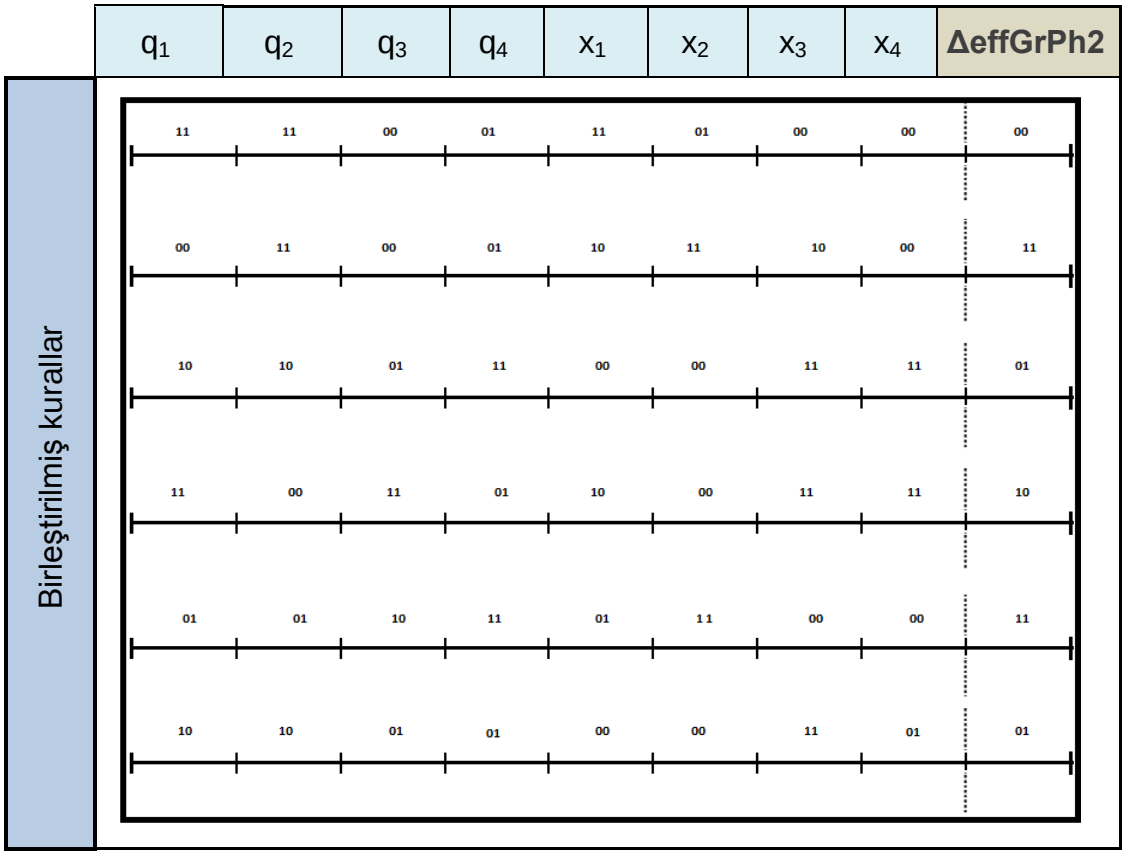


Şekil-5.1 Kavşağın toplam kuyruk miktarının 20 döngü boyunca artış grafiği

Yirmi adet döngü sonunda, kavşağın gider değeri 11000 araç civarındadır. GA'nın başarılı olabilmesi için bir sonraki iterasyonunda, yirmi adet döngü zamanı sonunda, toplam araç miktarının bu değerden daha düşük olması beklenmektedir.

Çizelge-5.3'de altı adet kuralın birleştirilmesi ile oluşturulmuş bir çözüm kümesi bulunmaktadır. Oluşturulan bu çözüm kümesi GA'nın bir kromozomu olarak ele alınmıştır. En uygun kromozom elde edilene kadar GA çalıştırılacak ve kromozomun genleri değiştirilecektir. Bir kromozomun iyi olup olmadığı uygunluk değerine bakılarak karar verilir. En iyi genetik koda sahip olan kromozomlar seçilecek; kötü genetiğe sahip kromozomlar ise havuzdan çıkartılacaktır. Böylelikle karmaşık çok boyutlu çözüm uzayında en uygun çözüm setleri bir araya getirilecektir.

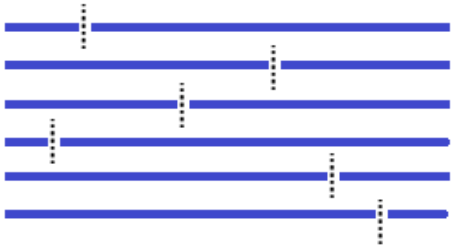
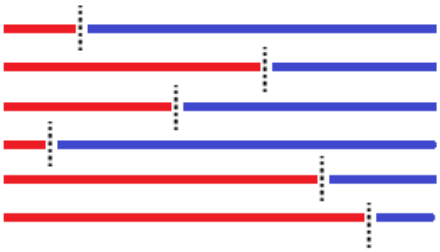
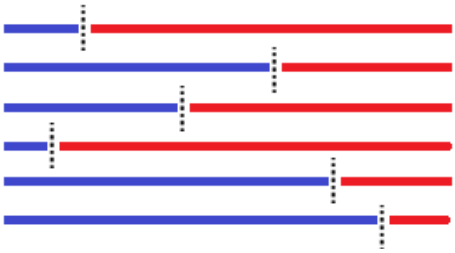
Çizelge-5.3 Altı adet kuralın bir araya gelerek oluşturduğu genetik algoritmanın bir kromozomu



5.4. Kromozomlar Arası Genetik Değişim (Çaprazlama, Crossover)

Elde edilen kromozom ve benzerlerinden başlangıç için rasgele kromozomlar üretilmiş ve bir kromozom havuzu yaratılmıştır. Rasgele üretilen kromozomların her biri sistemde test edililerek bir uygunluk değeri elde edilmiş ve elde edilen bu değer ile her biri işaretlenmiştir. Daha sonra işaretlenen kromozomlar uygunluk değerlerine göre sıralanmış ve uygunluk değeri en iyi olan iki kromozom alınarak bir sonraki kromozom havuzuna üzerlerinde herhangi bir işlem yapılmadan eklenmiştir. Seçkincilik (elitizm) olarak adlandırılan bu işlem, elde edilmiş en iyi sonucun korunmasını ve bir sonraki nesillere kaliteli genetiğin aktarılmasını garantileyen bir yöntemdir.

Çizelge-5.4 Çaprazlama işleminin şematik gösterimi

	Kromozomun Temsili Şekli	İkil Gösterimi
kromozom1		1101 | 00100111001010 1101101110 | 01100011 1001111 | 00011110111 101 | 010111100100111 010000100101 | 000111 11001100110001 | 1111
kromozom2		1101 | 11001001001001 1001011001 | 11000111 0011001 | 11000110001 010 | 011100011110011 011011110111 | 110011 11101101010110 | 1010
Birinci Çocuk		1101 | 11001001001001 1101101110 | 11000111 1001111 | 11000110001 101 | 011100011110011 010000100101 | 110011 11001100110001 | 1010
İkinci Çocuk		1101 | 00100111001010 1001011001 | 01100011 0011001 | 00011110111 010 | 010111100100111 011011110111 | 000111 11101101010110 | 1111

Bir sonraki aşama değerli bilginin yeni nesillere aktarılması ve kötü genetik materyale sahip kromozomların oluşturulan havuzdan atılması olacaktır.

İki kromozom çaprazlama işleminden geçirilirken aşağıda anlatılan yöntem izlenmiştir.

1) Rasgele oluşturulan nesilden (kromozom havuzundan) en kötü uygunluk değerine sahip iki kromozom havuzdan atılır
2) Geriye kalan nesilden en iyi uygunluk değerine sahip olan kromozomlar seçilir. Çaprazlama işleminden geçirilerek yeni nesil çocuklar (offspring)

elde edilir. En iyi kromozoma sahip olan bu iki kromozom oluşturulacak olan yeni nesile hiç bir mutasyona maruz kalmadan aktarılır.

3) Geriye kalan nesilden iki adet kromozom seçilir.
4) Kromozomları meydana getiren her bir kural tamamen rasgele bir yöntem ile bir noktadan ikiye ayrılır.
5) Rasgele bir noktadan kesilmiş olan kromozomun bir tarafında kalan genetik materyal ile diğer kromozomun ilgili yerleri birbirleriyle değiş tokuş yapılır
6) Meydana getirilen çocuklar havuza dahil olur ve sistemin gider değeri yeni nesil çocuklar ile tekrar hesaplanır.

Yapılan işlemlerin şematik gösterimi Çizelge-5.4'de gösterilmiştir.

Çaprazlama işlemi gerçekleştirildikten sonra meydana gelen çocuklara bir miktar mutasyon eklenmiştir. Böylelikle eğer optimizasyon yerel bir çözümüne takılırsa bu noktadan dışarı çıkıp evrensel çözüme ulaşabilme şansı artırılmış olur.

Çizelge-5.5 Mutasyon

Mutasyona uğramış birinci çocuk(offspring-1)	1101110010**1**1001001 1101**0**0111011000111 1001111111**1**00110**1**1 101**1**0**1**100011110011 0100101001010**1**0011 1100111**1**01100011010
Mutasyona uğramış ikinci çocuk (offspring-2)	1**0**0100100111**0**01010 10000**0**100101100011 0011001000111111**1** **1**0**1**010111100100111 0110**0**10**1**0111000111 111011010101101110

Çizelge-5.5 de acık renkte gösterilen ikililer mutasyon sonucu değişmiş olan ikillerdir. Mutasyonunun her bir kromozomdaki etkisi %12-%15 civarında seçilmiştir fakat uygunluk değeri en iyi olan iki kromozoma herhangi bir mutasyona işlemi uygulanmamıştır. Böylelikle en iyi sonuca yaklaşan kromozomlar korunmuş, algoritmanın çözüme yakınsama hızı artırılmıştır.

Uygunluk değeri en kötü olan iki kromozom ise havuzdan tamamen çıkartılmış ve hiç bir işlemde onlardan gelen genetik materyal kullanılmamıştır. Böylelik oluşturmuş olduğumuz yeni kromozom havuzunda en iyi gene sahip kromozomlar ile bunların çiftleşmesi ve bir miktar mutasyona uğraması sonucu oluşturulmuş çocuk kromozomlar bulunmaktadır.

5.5. Performans Hesabı

Her bir kural bloğunun bir önceki kural bloğu ile entegre edebilmek için bir performans parametresine ihtiyaç vardır. Bu parametre ancak bir referans noktasına başvurarak belirlenebilir. Bu çalışmada referans noktası olarak sıradan bir kavşak seçilmiştir.

Her bir döngü zamanı sonunda sıradan kavşak ile kural tabanının çalıştığı kavşaktan ayrılan toplam araç sayısı ihtiyaç duyulan performans değerinin hesaplanmasında kullanılmıştır.

Her döngü başında sıradan kavşak ile kural tabanlı kavşağın başlangıç koşulları bir olmayacaktır. Bu da karşılaştırma yapılırken yanlış sonuçların alınmasına neden olabilir. Bu yüzden karşılaştırma yapılacak olan döngü zamanlarında başlangıç koşulları birbirlerine eşitlenmiş, her bir yöndeki kuyruk miktarı ve her bir yönden gelen araç yoğunluğu eşit olacak şekilde ayarlanmış, ve performans hesabı aşağıda gösterildiği şekilde hesaplanmıştır.

$$\Delta = \Re(c) - \boldsymbol{\mathcal{S}}(\boldsymbol{c})$$

$$P = \frac{\Delta}{\boldsymbol{\mathcal{S}}(\boldsymbol{c})} + b \qquad (10)$$

Δ : Kural tabanlı çalışan kavşaktan ayrılan araç sayısı ile standart kavşaktan ayrılan araç sayısı arasındaki fark

$\Re$: Kural tabanlı kavşaktan c döngü zamanı sonucunda ayrılan toplam araç miktarı

$\boldsymbol{\mathcal{S}}$: Standart kavşaktan c döngü zamanı sonunda ayrılan toplam araç miktarı

b : $Denge\ değeri \sim 0.5$

P : $Performans\ değeri$

Performans hesabı her bir döngü zamanı sonunda yapılarak noktasal bir kontrol mekanizması geliştirilmiştir. Böylelikle performansın kötü olduğu yerler kullanıcı tarfından düzeltilebilecektir. Performansın kötü olduğu bir başka değişle sıradan kavşağın daha başarılı olduğu döngü sonlarında çalışan kurallar ve kuralların vermiş olduğu kararlar kullanıcı tarafından yeniden tasarlanarak performans artırılabilir. Böylelikle N adet döngü sonunda bulunan gider değeri küçültülebilir.

Denge değeri başlangıçta yaklaşık olarak 0.5 alınmıştır (10). Bu değer artırılırsa algoritma birleştirme işlemi yaparken eski sonuçları ağırlıklandırmakta, azaltıldığında ise yeni oluşturulan katmanın sonuçlarının etkisini ağırlıklandırmaktadır.

5.6. Kaynaştırma İşlemi

Performans değeri elde edildikten sonra en iyi sonucu vermiş olan kural bloğunun kararı, bir önceki kural bloğunun kararı ile birleştirilmelidir. Birleştirme işi her bir döngü sonunda kavşağın bir önceki kural bloğunun performansına bakılarak yapılır. Eğer performans bir önceki blokta iyi ise o bloğun vermiş olduğu karar; değilse yeni kural bloğunun verdiği karar ağırlıklandırılarak yeni performans değeri bulunur.

İki kural bloğunun vermiş olduğu karar aşağıdaki şekilde birleştirme işleminden geçirilmektedir.

$$\mathcal{D}(c) = \mathcal{F}(q, x, r)$$

$$\mathcal{D}_a(c) = \left(P_{l-1}(c) * \mathcal{D}_{l-1}(c)\right) + \left(1 - P_{l-1}(c)\right) * \mathcal{D}(c) \quad (11)$$

$\boldsymbol{q}$: Trafik akış oranı,

$\boldsymbol{x}$: Her yöndeki kuyruk yoğunluğu,

$\boldsymbol{r}$: Kural bloğu,

$\boldsymbol{\mathcal{F}}$: Durulaştırma fonksiyonu ya da karar fonksiyonu,

$\boldsymbol{\mathcal{D}}$: Karar, etkin yeşil süresindeki değişim oranı,

$\boldsymbol{P_{l-1}}$: Bir önceki bloğun c döngüsü zamanındaki performans değeri,

$\boldsymbol{\mathcal{D}_{l-1}}$: Bir önceki bloğun c döngüsü zamanındaki asıl kararı,

$\boldsymbol{\mathcal{D}_a}$: Asıl karar, uygulanacak asıl değişim oranı.

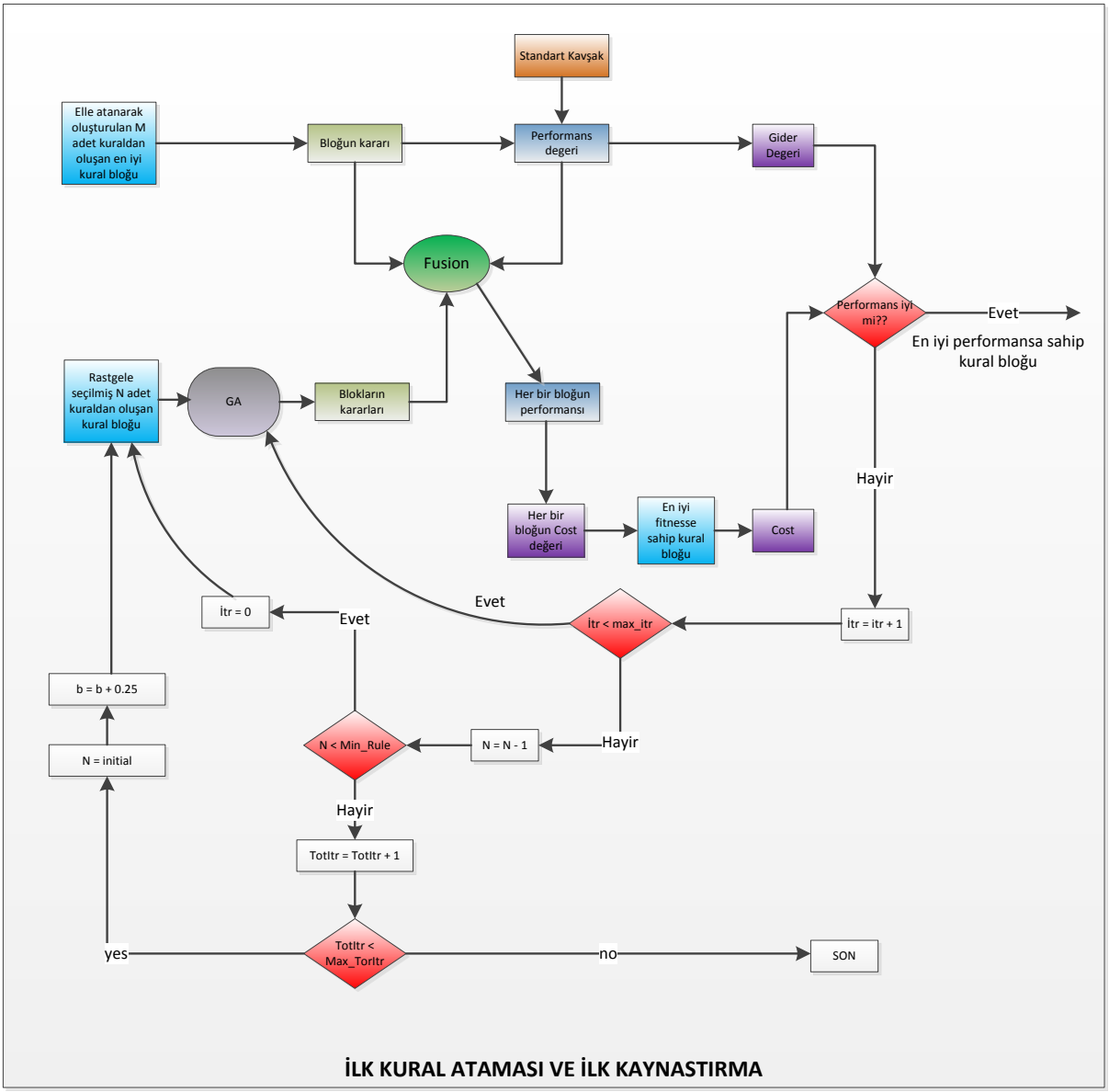
5.7. Algoritma

Çözüm için geliştirlen optimizasyon algoritmanın yazınsal gösterimi Ek-1 de gösterilmektedir.

5.8. Algoritmanın Akış Şeması

Ek-1'de anlatılan algoritmanın akış şeması ve ilk katmanın oluşturulması Şekil-6.1'de gösterilmektedir.

Şekil-5.2. Algoritmanın akış şeması



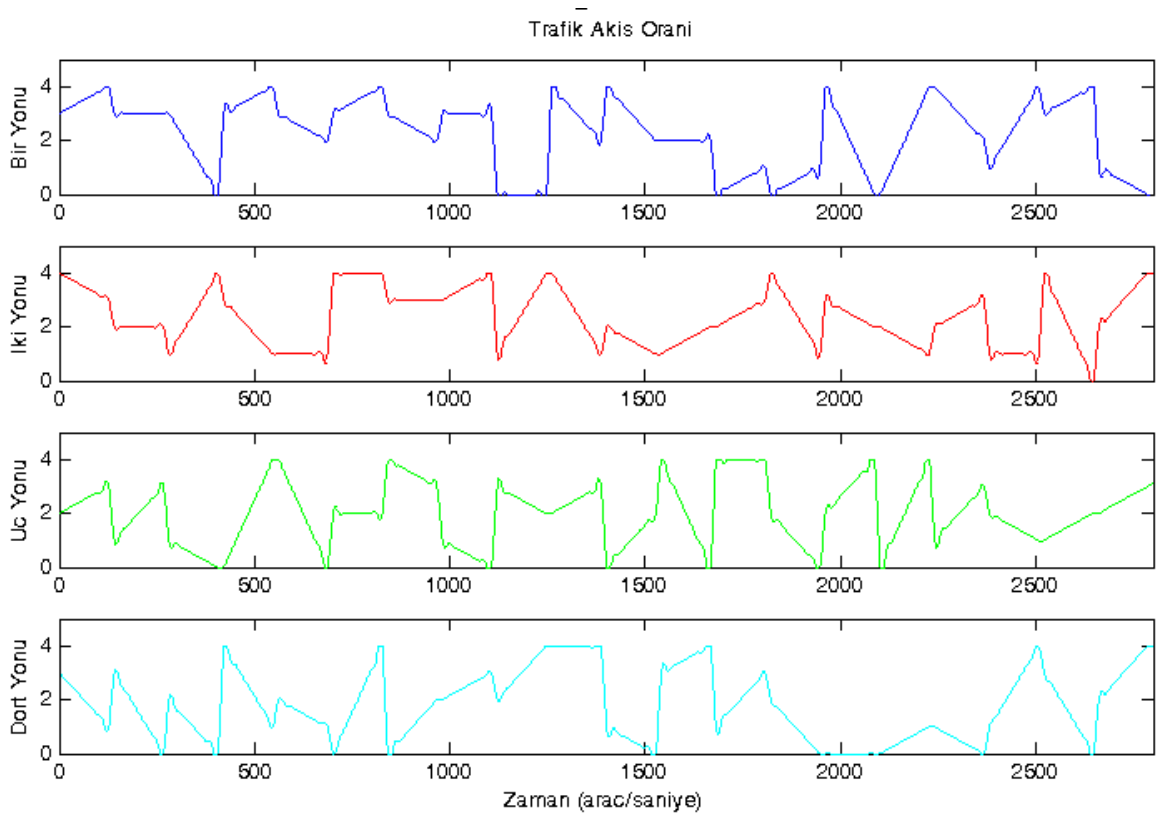
6. KAYNAŞTIRMA İŞLEMİ VE SONUÇLARI

6.1. Eğitim Verisi

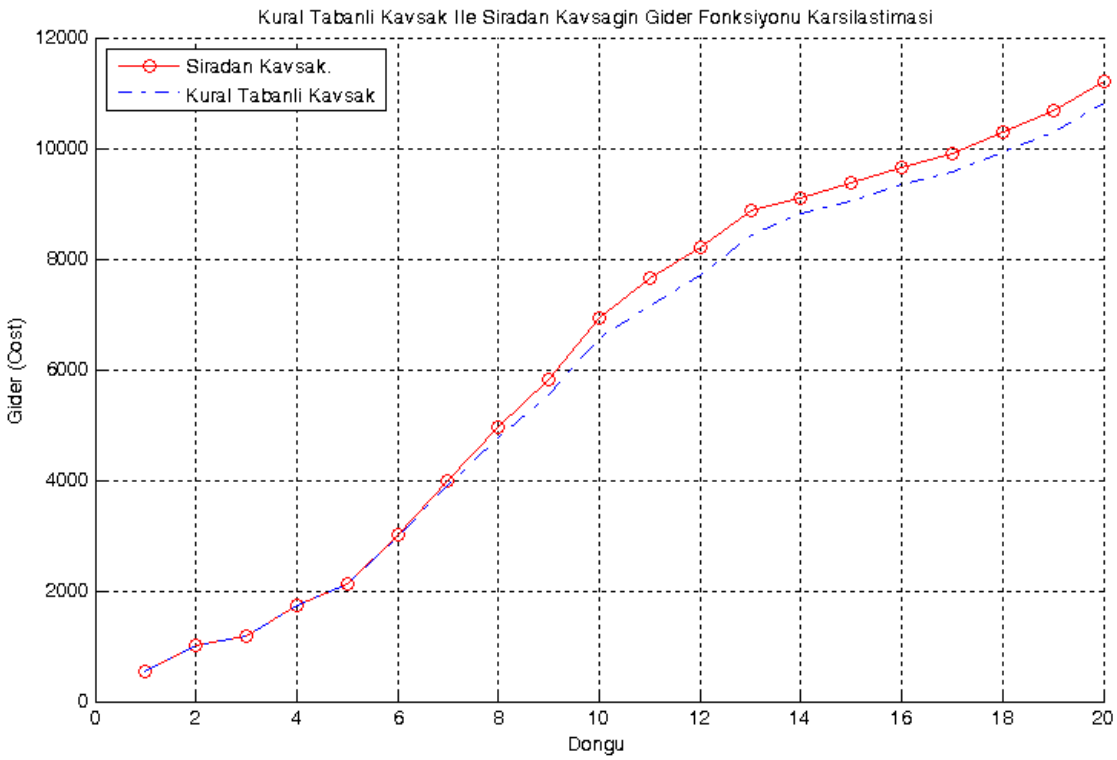
Tüm durumlara en uygun etkin yeşil sürelerinin ataması yapılabilmesi için eğitim verisinin çok kapsamlı oluşturulması gerekmektedir. Her bir başlangıç durum senaryosunun olduğu bir veri seti ile eğitilmiş kurallar, test verileri ile denendiğinde, en doğru çözüme ulaşacaktır.

Eğitim verisi oluşturulurken sürekli olasılık dağılım ailesinden olan Gaussian dağılım kullanılmıştır [21]. Böylelikle birbirini tekrar eden durumlardan mümkün olduğu kadar uzak durulmaya çalışılmış ve olası tüm koşulların eğitim verisinin içerisinde olması sağlanmıştır.

Yirmi döngü zamanlık bir deneme verisi geliştirdiğimiz çözüm metodu için yeterli olup her bir yön için Şekil-6.1'deki gibi oluşturulmuştur. Her bir döngü zamanı 140 saniye olacak şekilde düşünülmüştür. Eğitim verisinin uzunluğu 140*20 = 2800 adet veriden oluşmaktadır. Bir başka anlatımla eğitim verisinin her bir girdisi saniyede gelen araç miktarını temsil etmektedir.



Şekil-6.1 Eğitim Verileri



Şekil-6.2 Kural tabanlı çalışan kavşak (kesik çizgili eğri) ile sıradan kavşağın (yuvarlak çizgili eğri) gider fonksiyonu karşılaştırması

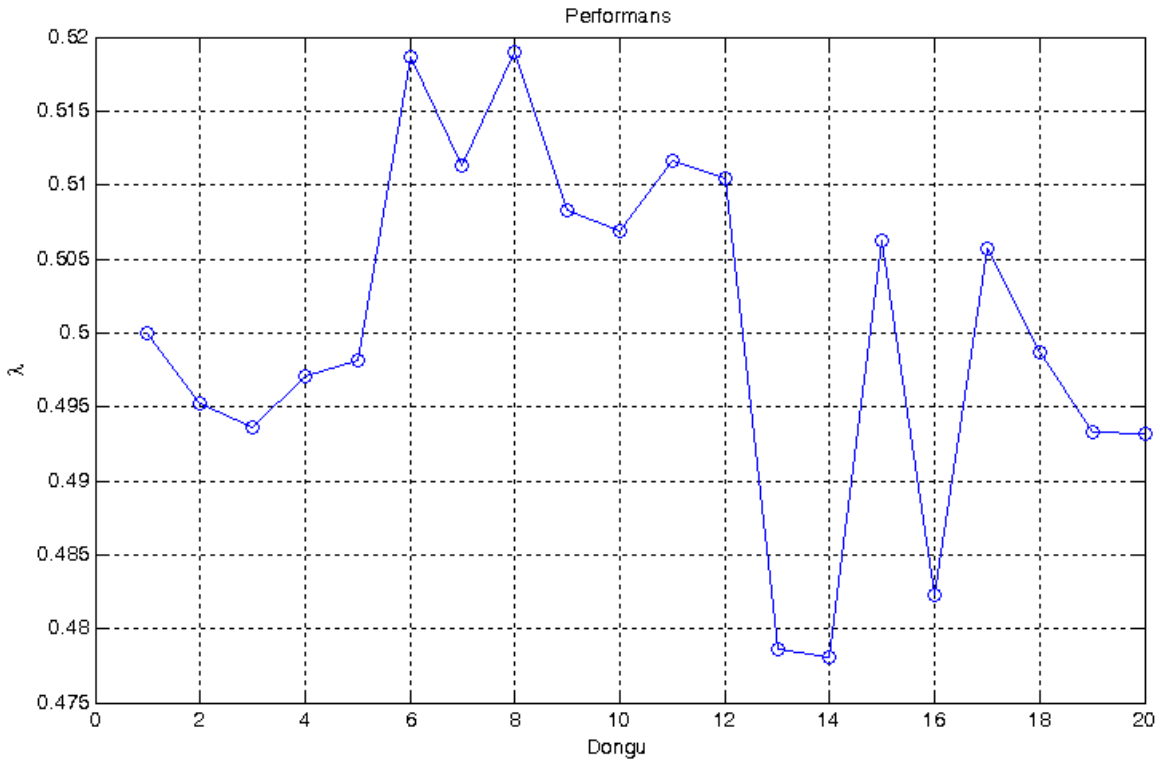
Oluşturulan eğitim verisinin çok yönlü olması sağlandıktan sonra kural tabanlı kavşak bu veriler ile eğitilmiştir

Her bir yön için oluşturulan araç yoğunluğu birbirinden farklı olduğu için her bir yönde oluşacak kuyruklanma miktarları da farklı olacaktır. Böylelikle farklı kuyruk uzunluklarında kavşağın en uygun kararı vermesi için kurallar eğitilebilmiştir.

6.2. Kaynaştırma İşleminin Verime Etkisi

Başlangıçta “en iyi karar” ataması olacak şekilde tasarlanan kuralların oluşturduğu gider fonksiyonun grafiği Şekil-6.2’deki gibidir. En iyi kararlar seçildiği için başlangıç aşamasında kural tabanlı çalışan kavşağının veriminin sıradan kavşaktan daha başarılı olması beklenen bir sonuçtur.

Her bir döngü zamanı sonunda hesaplanan kuyruk miktarı, bir sonraki döngü zamanında hesaplanan kuyruk miktarına eklenerek toplam artış elde edilmiştir. Bu artış ne kadar yavaş olursa, yani grafiğin altında kalan alan ne kadar az ise, kural tabanlı kavşağın sıradan kavşağa göre başarımı o kadar fazla olacaktır.



Şekil-6.3 Kural tabanlı kavşak ile sıradan kavşağın karşılaştırılması sonu elde edilen performans değeri

Şekil-6.2'de kesik çizgili eğri ile gösterilen kural tabanlı kavşağın gider fonksiyonunu, yuvarlak çizgili eğri ile gösterilen sıradan kavşağın gider fonksiyonunu göstermektedir. Gider fonksiyonları incelendiğinde başlangıçta her iki kavşağın veriminin eşit olduğu zamanla kural tabanlı çalışan kavşağın daha fazla miktarda aracın kavşaktan ayrılmasına imkan tanıyarak kavşağın performansını artırdığı görülmektedir. Olası durumlar daha önceden tahmin edileceği için yeşil süresindeki atamada ona göre olacaktır. Bu örnekte başlangıçta 30 adet kural kullanıcı tarafından karar kısmı en iyi olacak şekilde tasarlanmış ve daha sonra GA vasıtası ile kurallar eğitim verilerine göre eğitilmişlerdir. Kurallar eğitilirken bir yandan da her bir döngü zamanı sonunda kural tabanlı kavşak ile sıradan kavşaktan ayrılan araç miktarı hesaplanmış bir performans değeri grafiği elde edilmiştir (Şekil-6.3).

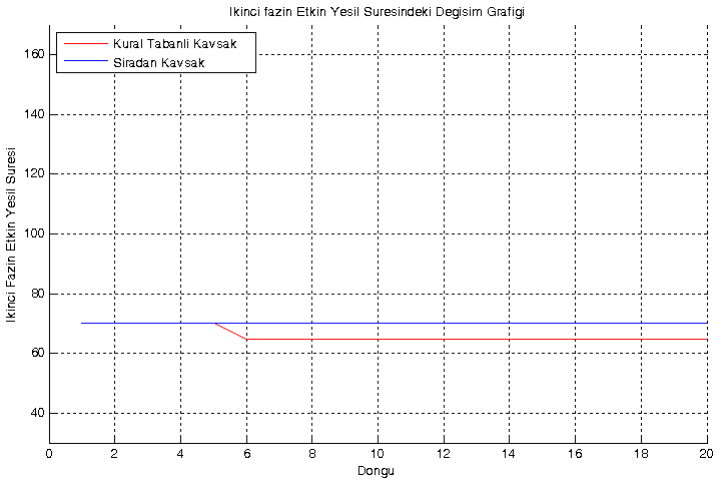
Her iki kavşağada döngü zamanları boyunca aynı miktarda araç gelmiştir. Performans hesabı yapılırken haksızlık olmaması için her iki kavşağında başlangıç koşullarının aynı olması gerekmektedir. Bu yüzden sıradan kavşağın başlangıç koşulu kural tabanlı kavşağın başlangıç durumu ile aynı hale getirilmiş

ve performans değeri hesaplanmıştır. Böylelikle kural tabanlı kavşağın, belirli bir trafik yoğunluğunda standart kavşağa göre ne kadar performanslı çalıştığı bulunmuştur.

Performans hesabı yapılırken (11) nolu denklemde kullanılan denge değeri 0.5 olarak seçilmiştir. 0.5 değerinin altında kalan yerler kural tabanlı çalışan kavşağın kötü çalıştığı yukarısında kalan yerler ise iyi çalıştığı durumları göstermektedir. Şekil-6.3 incelendiğinde kural tabalı çalışan kavşak, sıradan kavşağa göre sıfır ile beşinci döngü zamanları arasında kötü çalışmış, beş ile on üçüncü döngü zamanları arasında iyi çalışmış, geri kalan döngü zamanlarında kısmen iyi kısmen kötü çalışmıştır. Bunun nedeni başlangıçta seçilen 30 adet kuralın bu noktarlarda yeterli performansı gösterememesidir. Genetik Algoritma her bir noktaki performansı iyileştirmek yerine 20 adet döngü sonundaki toplam araç miktarını azaltmaya çalıştığı için kuralları bu duruma göre eğitmektedir. Bu durum bazı döngü sonlarında performasın kötüleşmesine neden olsa da yirmi adet döngü sonunda gider değerini azaltmaktadır.

Kural tabanlı kavşağın ilk atamaya göre gayet başarılı çalıştığı kabul edilebilir. Nihayetinde sistemin 3^8 farklı durumu vardır. Kural tabanlı kavşak ise sadece 30 adet kuralın mantıklı seçimi ile sıradan kavşaktan daha başarılı olmuştur.

Bu noktada kural tabanlı kavşağın etkin yeşil süresindeki değişimi incelemek faydalı olacaktır (Şekil-6.4). Örnek olarak seçilen uygulamada bir döngü zamanı 140 saniye olarak belirlenmiştir. Sıradan kavşağın etkin süresi sabit ve 70 saniye olarak düşünülmüştür.

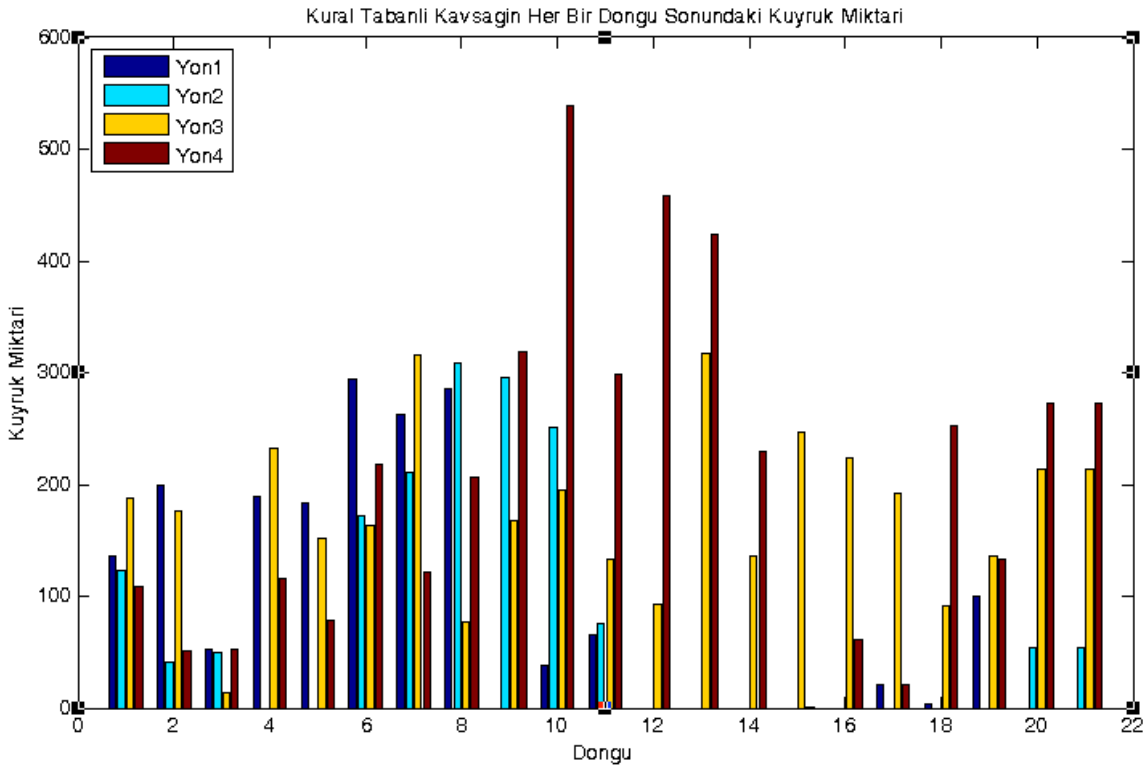


Şekil-6.4 Sıradan kavşak ile kural tabanlı kavşağın yeşil sürelerindeki değişim

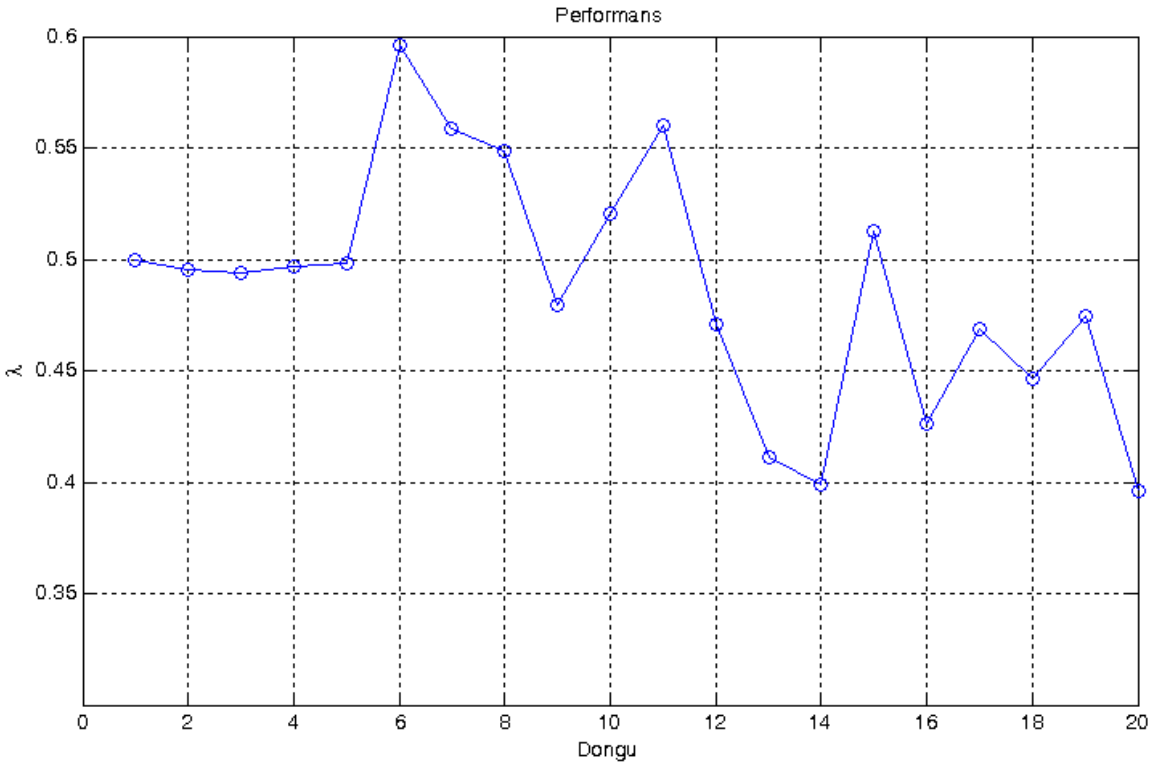
Şekil-6.4 de gösterilen örnekte kural tabalı çalışan kavşağın etkin yeşil süresi koyu renk ile gösterilirken sıradan kavşağın etkin yeşil süresi açık renk ile gösterilmiştir. Grafik incelendiğinde görülecektir ki kural tabanlı kavşağın ikinci fazının etkin yeşil süresi beş nolu döngünün sonunda 70 saniyeden 64 saniye civarına düşmüştür. Bu değerden sonra herhangi bir değişim olmamıştır.

Şekil-6.1'deki eğitim verisi incelendiğinde beş nolu döngünün başında (yaklaşık olarak 700 saniye civarında) bir nolu faz yönünde araç yoğunluğunda bir artış olmaktadır. Bu durumda kural tabanlı çalışan kavşak ikinci fazın etkin yeşil süresini biraz düşürerek (bu örnekte altı saniye kadar azaltmıştır) birinci fazın yeşil süresini artırmıştır. Bu durumda beşinci döngüden on üçüncü döngüye kadar kural tabanlı kavşağın (8) nolu denklemde gösterildiği üzere uygunluk değerinin artmasına neden olmuştur.

Eğitim verisinin 13'üncü ve 14'üncü döngü zamanları incelenecek olursa, burada tam tersi bir durum gerçekleştiği kolayca görülebilir fakat bu duruma kural tabanlı kavşak yeterli bir çözüm yaratamamıştır. Bu nokta yeni kurallar kaynaştırma metodu ile kaynaştırıldığında sonuçun daha iyi olduğu ilerleyen kısılarda gösterilmiştir.



Şekil-6.5 Her bir döngü sonunda kural tabanlı kavşağın kuyruk miktarı



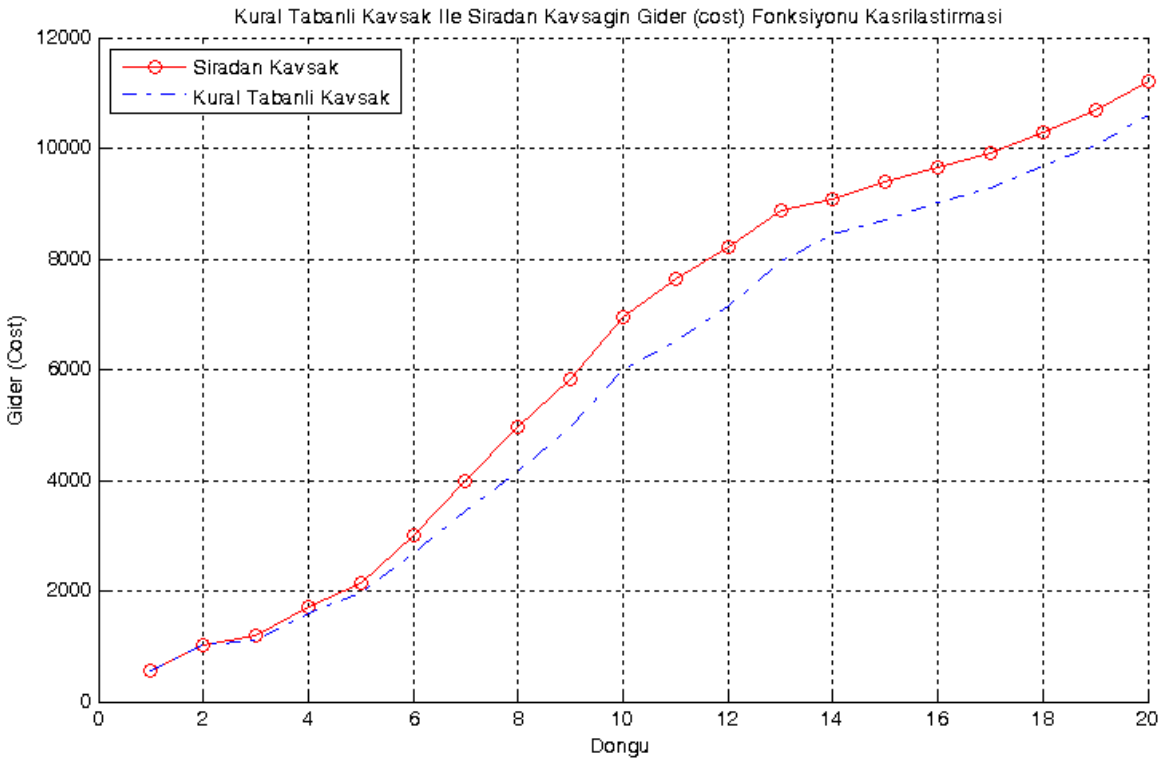
Şekii-6.6 Birinci kaynaştırma işlemi sonucu perfomans grafiği

Kural tabanlı kavşağın çalışmasında bir başka unsur olan kuyruk miktarının değişim grafiği Şekil-6.5'de gösterilmiştir. Grafikte her bir yöndeki kuyruk miktarı farklı renklerle gösterilmiş olup her bir kolon döngü sonundaki araç miktarı ifade etmektedir.

Grafik incelendiğinde görülecektir ki kural tabanlı çalışan kavşağın kuyruk miktarındaki dikkate değer değişiklik beşinci döngü sonunda gerçekleşmiştir. Bu noktada kural tabanlı kavşağın etkin yeşil süresi 70 saniyeden 64 saniyeye düşmüştür(Şekil-6.4). Beşinci döngüden sonra birinci fazın etkin yeşil süresi arttığı için bir nolu yönde ve iki nolu yönde (grafikte Line1 ve Line2) kuyruk miktarı hızla artmış, diğer yönlerdeki (grafikte Line3 ve Line4) kuyruklanma azalmıştır. Toplam kuyruklanma miktarı ise Çizelge-6.2'de gösterildiği gibi hep azalış göstermiştir.

6.2.1. Birinci Kaynaştırma İşlemi

Kaynaştırma işlemi gerçekleştirilirken performansın iyi olduğu yerlerde eski katmanın vermiş olduğu kararların etkisi, performansın kötü olduğu yerlerde ise yeni kuralların vereceği kararların etkisi daha baskın seçilmiştir. Bu işlem için (10) nolu denklemde gösterilen denge değeri kullanılmıştır.



Şekil-6.7 Birinci kaynaştırma sonrası gider fonksiyonu grafiği

Rasgele oluşturulan kurallara ilk kaynaştırma işlemi yapıldıktan sonra kural tabanlı kavşağın performans değerindeki değişim Şekil-6.6'daki gibi olmuştur.

Grafik incelendiğinde görülecektir ki bir önceki katmanın performansı sıfır ile beşinci döngüler arası kötüyken yeni kuralların kaynaştırılması sonucunda bu noktalarda bir iyileşe olmuştur, fakat hala sıradan kavşağa göre çok başarılı değildir. Beşinci ile sekizinci döngüler arasında ise iyi olan performans daha da iyiye gitmiş ve neredeyse 0.1 kat daha fazla adette aracın kavşaktan ayrılmasına olanak tanımıştır. Birinci kaynaştırma işlemi sonucu kavşağın gider fonksiyonu grafiği Şekil-6.7 de gösterildiği gibidir. Şekil-6.7 incelendiğinde görülecektir ki kural tabanlı kavşağın gider fonksiyonu altında kalan alan ilk atamada altında kalan alandan daha küçüktür.

Burada dikkat edilmesi gereken bir nokta vardır. Toplam döngü sonunda iyileşme sağlayan bu yapı bazı noktalarda performans kaybına neden olmaktadır. Örneğin dokuzuncu döngü zamanını inceleyecek olursak bir önceki katmanda bu noktada daha iyi bir performans sağlanıyorken, kaynaştırma işlemi yapıldıktan sonra bu noktada performans düşmüştür. Kural tabanlı çalışan kavşak toplamda daha

düşük gider elde etmeye çalıştığı için bazı noktalarda performans kaybının oluşmasına neden olmaktadır.

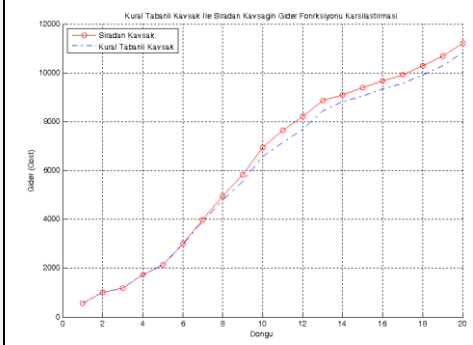
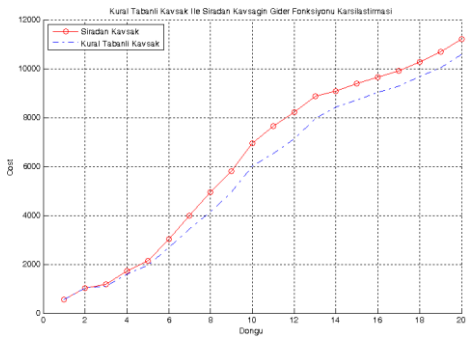
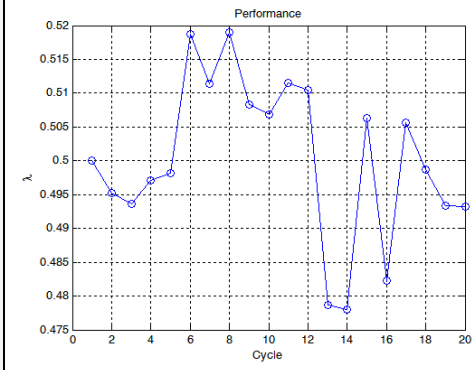
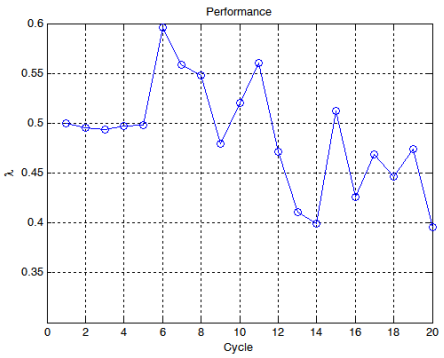
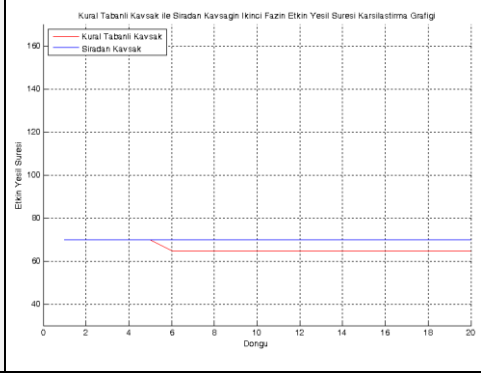
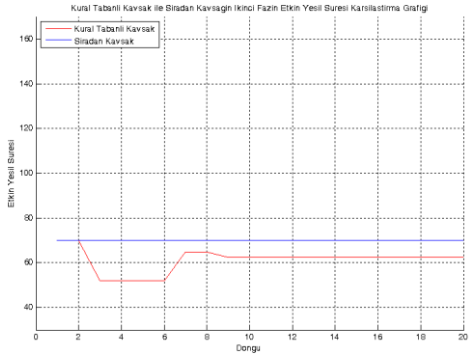
6.2.2. İki ve Daha Fazla Kaynaştırma İşlemi

Kavşağın performasının artırılabilmesi için yeni kural katmanlarının yaratılması, eğitilmesi, iki ve daha fazla sayıda kaynaştırma işleminden geçirilmesi gerekmektedir. Bu çalışmada çok detaylı bilgi barındıran eğitim datasını işleyebilmek için beş adet kaynaştırma işleminin uygulanması yeterli olmuştur. Her bir kaynaştırma işlemi öncesi 30 adet kural rasgele oluşturulmuş, karar kısımları rastgele atanmış ve GA'nın yardımıyla uygunluk değeri en iyi oluncaya kadar algoritma çalıştırılmıştır. Bütün bu işlemlerin sonucunda kural tabanlı çalışan kavaşak sıradan kavşaktan %20-30 arası daha başarılı bir performans çizmiştir.

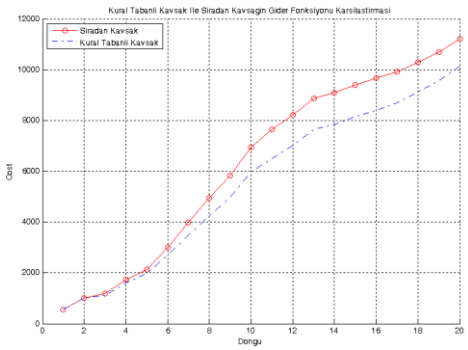
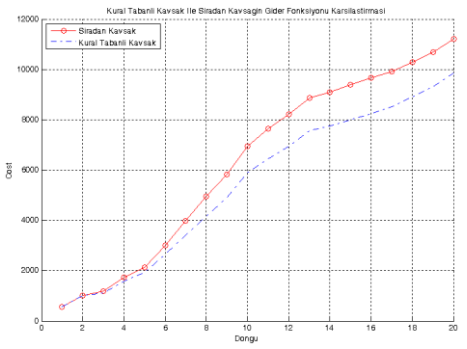
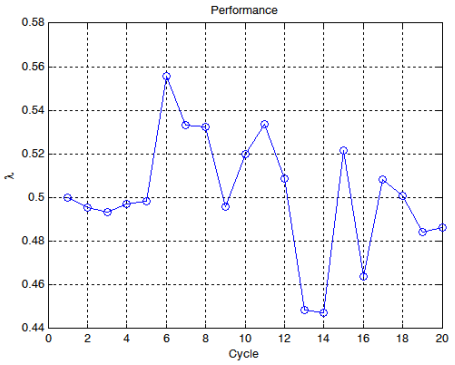
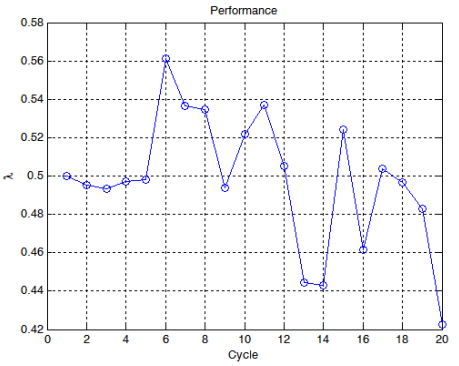
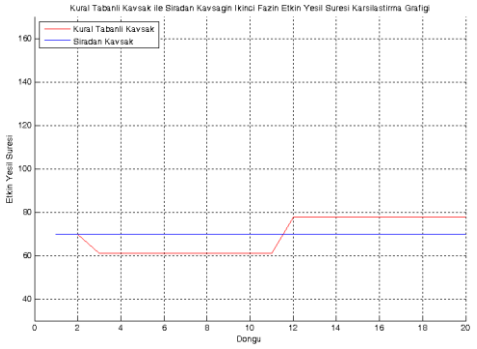
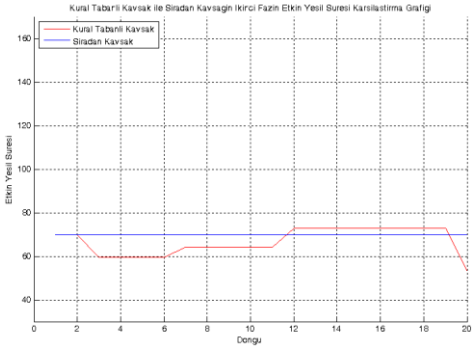
Çizelge-6.1'de beş adet kaynaştırma işlemi sonucu elde edilen gider fonksiyonu, performans ve ikinci fazın etkin yeşil süresindeki değişim grafiği görülmektedir.

Çizelge-6.1

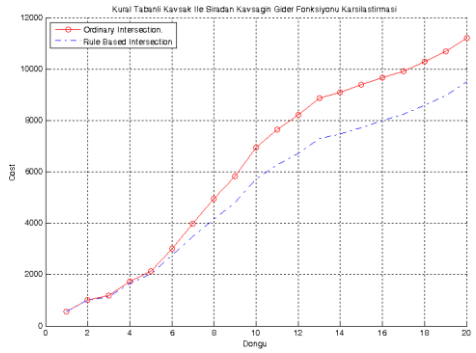
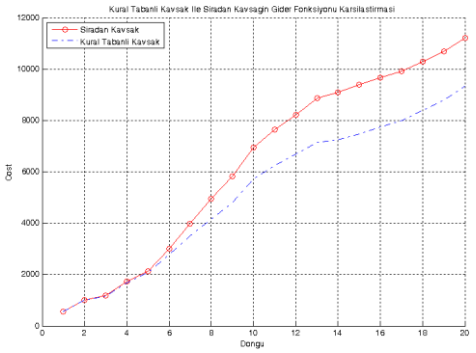
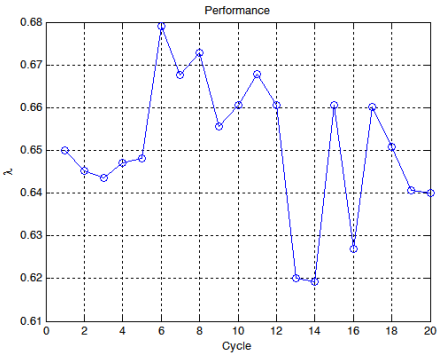
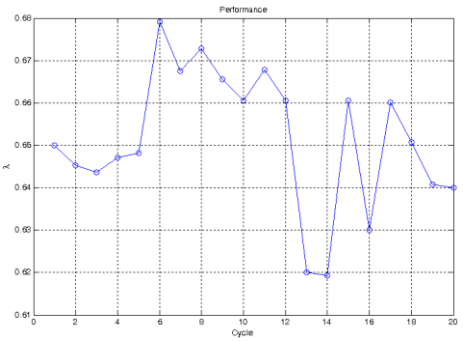
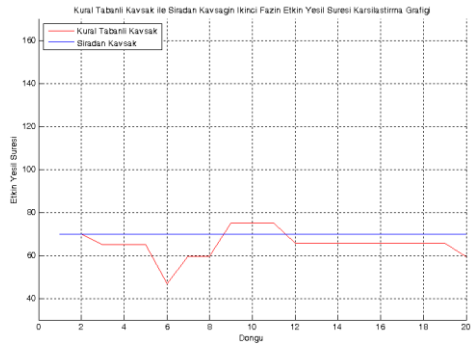
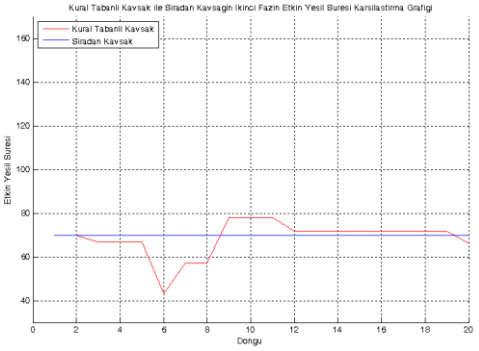
a) İlk atama Ve birinci Kaynaştıma İşlemi Sonuçları

	İlk Atama	Birinci Kaynaştırma
Gider Fonksiyonu Grafiği		
Performans Grafiği		
Faz-2 Etkin Yeşil Süresi		

b) *Üçüncü* ve dördüncü Kaynaştıma İşlemi Sonuçları

	İkinci Kaynaştırma	Üçüncü Kaynaştırma
Gider Fonksiyonu Grafiği		
Performans Grafiği		
Faz-2 Etkin Yeşil Süresi		

c) *dördüncü* ve beşinci Kaynaştıma İşlemi Sonuçları

	Dördüncü Kaynaştırma	Beşinci Kaynaştırma
Gider Fonksiyonu Grafiği		
Performans Grafiği		
Faz-2 Etkin Yeşil Süresi		

6.3. Eğitilen Kural Bloklarının Test Verilerine Uygulanması

Eğitilen kural bloklarının gerçek veriler ile test edilebilmesi için performans verisinden yararlanmak gerekmektedir. Performans verisinden yararlanabilmek için de her bir döngü zamanı sonunda kavşağın durumu ile eğitim verileri kullanılırken oluşan durum arasında bir benzerlik kurmak gerekmektedir. Bu benzerlik aşağıdaki denklem ile sağlanmıştır.

$$d_i = ||\vec{u} - \overrightarrow{u_i}||$$

$$w_i = \begin{cases} \frac{1}{d_i + \in}, & eğer\ d_i \\ \in, & eğer\ d_i \end{cases}$$

$$p_a = \frac{\sum_{i=0}^{N} w_i * p_i}{\sum_{i=0}^{N} w_i} \qquad (12)$$

i : Döngü indisi

$\overrightarrow{u_i}$: Eğitim verileri işlenirken döngü sonundaki kavşağın durumu

$\vec{u}$: Test verileri işlenirken döngü sonundaki kavşağın durumu

d_i: Test esnasında kavşağın döngü sonundaki durumunun eğitim esnasında elde edilen döngü sonundaki kavşağın durumuna benzerliği, mesafe vektörü

p_i: Eğitim esnasında bulunan döngü sonundaki performans değeri

p_a: Test esnasında kullanılacak performans değeri

Bu işelem özetle kural tabanlı kavşağın döngü sonundaki durumununun, eğitim verleri kullanılırken oluşan döngü sonundaki duruma uzaklığını bulmaktadır. Uzaklık ile ters orantılı olarak yeni performans değerlerini hesaplamaktadır.

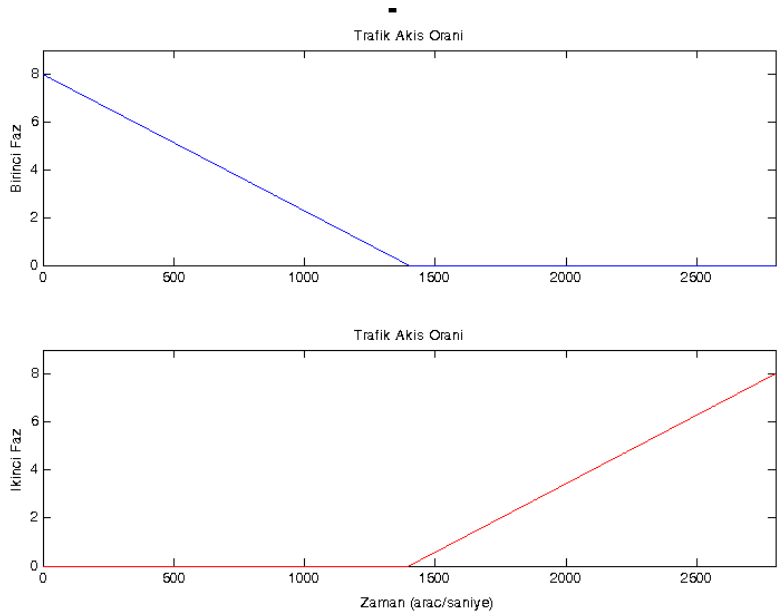
7. SONUÇLAR

7.1. Test Verileri İle Yapılan Çalışmaların Sonuçları

Kural blokları en uygun sonucu bulacak şekilde eğitilip, her bir döngü sonundaki perfomans değerleri bulunduktan sonra elde edilen kurallar test verilerine uygulanmıştır. Bir sonraki bölümde farklı test verileri oluşturulmuş ve eğitilen kurallar bu test verileri ile test edilmiştir. Test verileri elde edilirken her bir yönün araç yoğunluğu yerine birinci ve ikinci fazın toplam araç yoğunlu ele alınmıştır. Karşılaştırma yapabilmek için sıradan kavşak ile kural tabanlı kavşağın başlangıç koşulları eşitlenmiş ve gider fonksiyonları grafiği incelenerek başarım karşılaştırılması yapılmıştır.

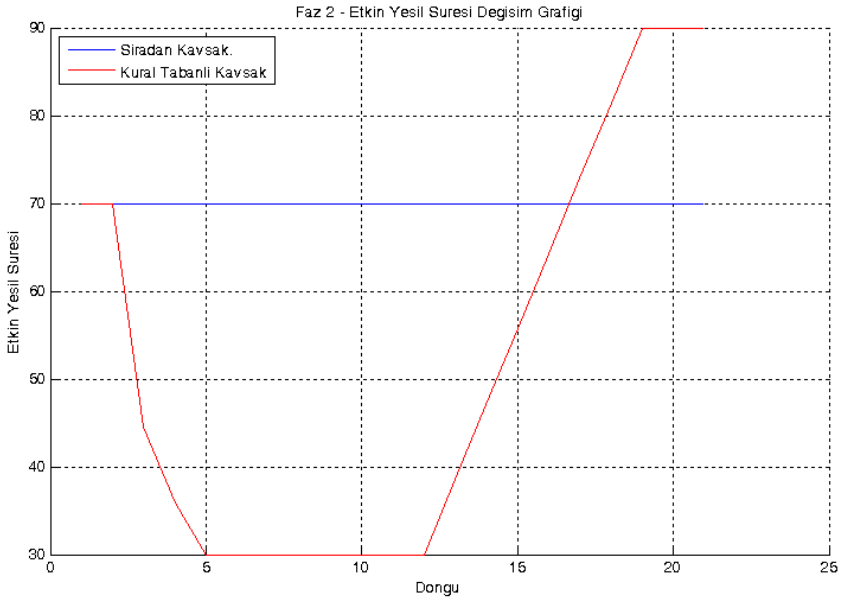
7.1.1. Birinci Deneme Sonuçları ve Açıklamalar

Birinci deneme verisinde kuralların işleyişini gözlemleyebilmek için iki fazın araç yoğunluğu zıt olacak şekilde ayarlanmıştır. Birinci fazda 1400 saniye boyunca araç varken ikinci faza hiç araç gelmemiş, 1400 saniyeden sonrada tam tersi bir durum oluşturulmuştur (Şekil-7.1)



Şekil-7.1 Birinci deneme için oluşturulan bir ve iki nolu fazların araç yoğunluğu test verisi

Araç yoğunluğu Şekil-7.1'deki gibi seçildiğinde kural tabanlı kavşak ile sıradan kavşağın etkin yeşil sürelerindeki değişim grafiği Şekil-7.2'daki olmaktadır.

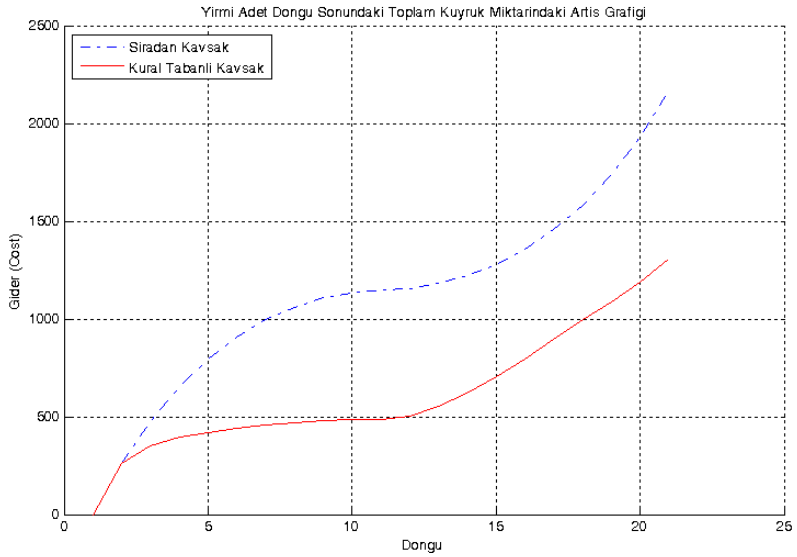


Şekil-7.2 Birinci denemede elde edilen etkin yeşil süresindeki değişim. (Sabit ve 70 olan sıradan kavşağın etkin yeşil süresini göstermektedir)

Şekil-7.2 da sabit ve mavi renk ile gösterilen grafik sıradan kavşağın etkin yeşil süresini gösterirken, diğer grafik kural tabanlı kavşağın yeşil süresindeki değişimini gösterir.

Şekil-7.1'deki test verisi incelendiğinde 1400 saniye boyunca birinci fazın araç yoğunlunun fazla olduğu, ikinci fazın araç yoğunluğunun ise hiç olmadığı görülmektedir. Bu durumda eğer kavşakta bir trafik polisi görevli olsaydı, 1400 saniye boyunca birinci faz yönündeki araçlara, Şekil-3.2'de gösterilen bir ve iki nolu yönlere, geçiş hakkı tanıyacağı; 1400 saniye sonrasında ise ikinci faza geçiş hakkı tanıyarak kavşakta araçların beklemesini tamamen ortadan kaldırabileceği kolayca gözükmektedir.

Kural tabanlı çalışan kavşağın burada uygulamaya çalıştığı yöntem de trafik polisinin yapmaya çalıştığı yönteme çok benzer olmuştur. Şekil-7.2 incelendiğinde birinci döngü sonunda ikinci fazın etkin yeşil süresini 70 saniyeden 45 saniye civarına indirmiş, ikinci döngü sonunda ise etkin yeşil süresini 30 saniye olan en az etkin yeşil süresi seviyesine indirerek birinci faz yönüne daha uzun süre geçiş hakkı tanınmasını sağlamıştır.



Şekil-7.3 Sıradan kavşak ile kural tabanlı kavşağın gider fonksiyonu karşılaştıması

Onbirinci döngü sonunda trafiğin araç yoğunlu tam tersine dönümüş ve faz iki yönünde araç yoğunluğu artmıştır. Bu durumda kural tabanlı çalışan kavşak duruma kendini uyarlayarak ikinci fazın süresini beş döngü sonunda en fazla etkin yeşil süresi olan 70 saniyeye seviyesine çıkartarak kavşakta gereksiz beklemelerin ve kuyruklanmanın önüne geçmiştir.

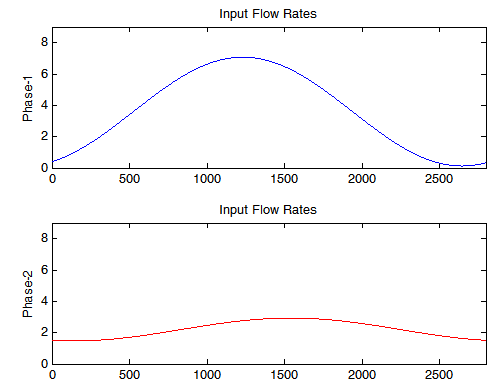
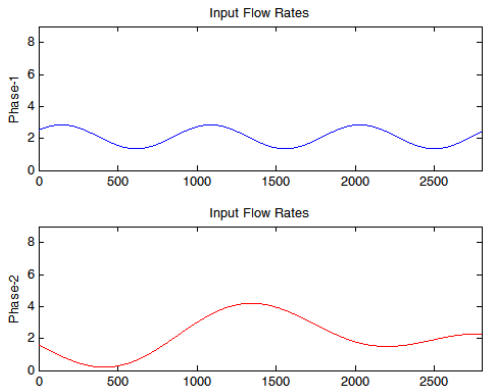
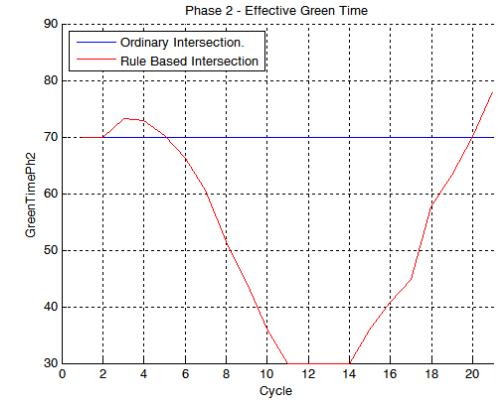
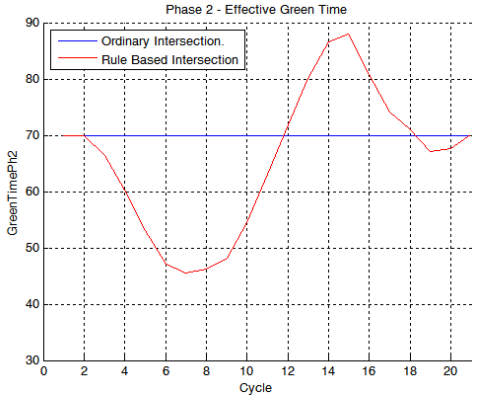
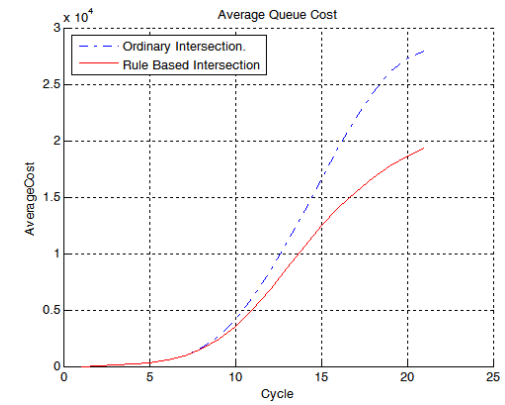
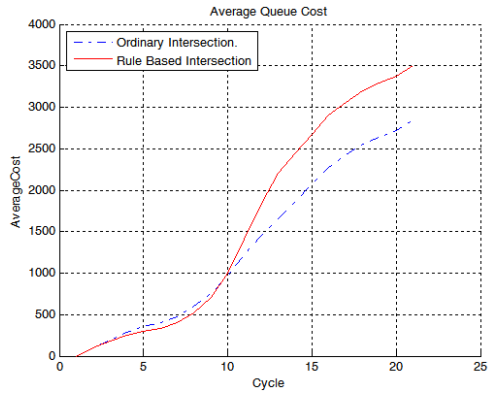
Şekil-7.3 incelendiğinde sıradan kavşak ile kural tabanlı kavşağın gider fonksiyonu grafiği karşılaştırması görülecektir. Şekil-7.1'de gösterilen test verileri ile eğitilmiş kurallar test edildiğinde uyarlamalı çalışan kavşağın sıradan kavşağa göre %40 başarı sağladığı görülebilir.

7.1.2. Farklı Test Verileri İle Yapılan Deney Sonuçları

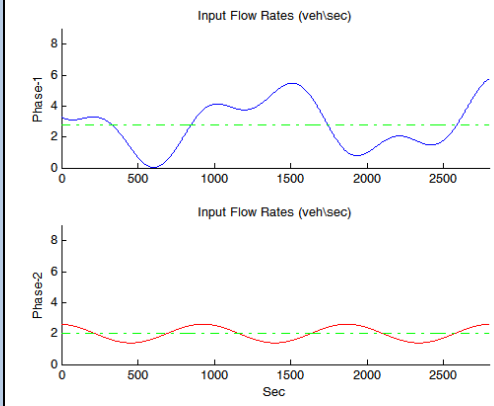
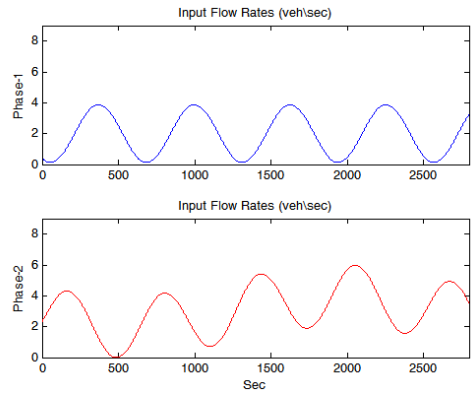
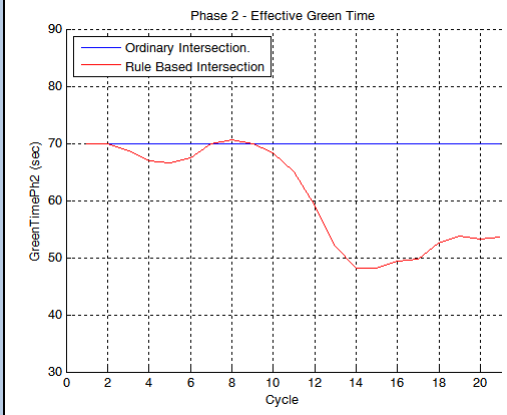
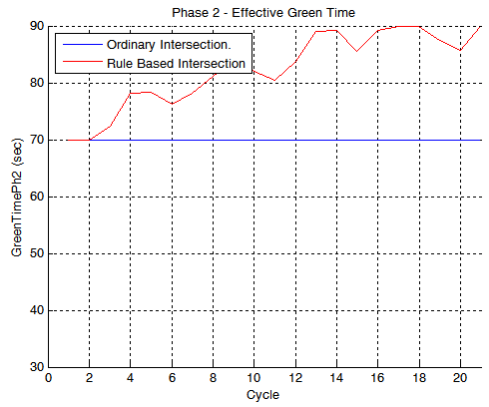
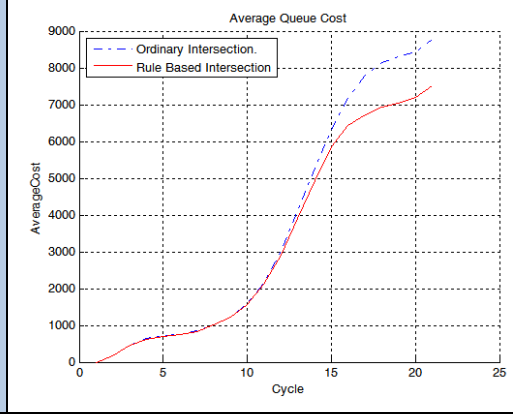
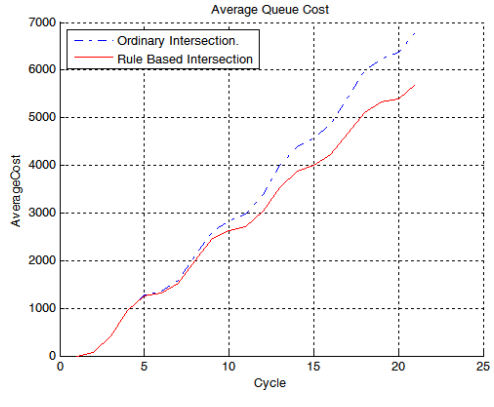
Elde edilen çözümün başka test verileri ile nasıl bir davranış sergilediği Çizelge-7.1'de gösterilmiştir.

Çizelge-7.1 Değişik Test Verileri ile Eğitilmiş Kuralların Test Edilmesi

a) Birinci ve İkinci Denemeler

	Deneme 1	Deneme 2
Test Verisi		
Faz 2 Etkin Yeşil Süresi		
Gider Fonksiyonu		

a) Üçüncü ve Dördüncü Denemeler

	Deneme 3	Deneme 4
Test Verisi		
Faz 2 Etkin Yeşil Süresi		
Gider Fonksiyonu		

7.2. Öneriler ve Tartışma

Bu çalışmada bir çok yolun kesiştiği yerlerde trafik sıkışıklığının önlenerek kavşağın daha etkin kullanılması amacına yönelik kural tabanlı bir kavşak modeli geliştirilmiştir. Literatürde yer alan çeşitli kontrolcü modelleri incelenmiş, bunlar arasında [17] nolu makalede kontolcü modeli geliştirilirken kullanılan durum denklemi, kural tabanlı çalışan kavşağın durum denkleminin oluşturulmasında kullanılmıştır.

Bir kentin tamamının trafik sıkışlığını önlemek için öncelikle her bir kavşağın kendi içerisinde trafik sıkışıklığını önlemesi, bir başka değişle her bir fazın etkin yeşil sürelerinin en iyi şekilde ayarlanması gerekmektedir. Kavşağın, trafik sıkışıklığını önleyecek şekilde ışık sürelerini ayarlayabilmesi için trafik yoğunluğunu, kavşakta biriken araç miktarını, algılayıcılar vasıtasıyla denetçisine aktarması ve bu duruma en uygun etkin yeşil sürelerinin denetçi tarafından belirlenmesi gerekmektedir. Böylelikle kavşak, döngü boyunca beklemeleri en az seviyede tutarken; kavşaktan ayrılan araç sayısını en fazla seviyede tutabilecektir.

Geliştirilen kural tabanlı kavşak modeli her bir olası trafik durumu için en uygun kararın verildiği bir modeldir. Karar bir önceki döngü zamanında kullanılan etkin yeşil süresinin ne kadar değiştirileceğini söyler. Etkin yeşil süresi bulunan karar oranında bir miktar artırılır ya da azaltılır.

Birden fazla kuralın bir araya getirilerek oluşturulan kural bloğu, kavşak modelinin kural katmanını oluşturmaktadır. Katmandaki her bir kural, kavşakta olası bir durumu temsil etmektedir. Birinci faz (Şekil-3.2'de gösterilen bir ve iki yönü) yönünde yoğun bir trafik olması ve bu yöndeki biriken kuyruk miktarının fazla olduğu durum kural tabanının öncül (premise) değişkenini meydana getirirken; ikinci fazdaki etkin yeşil süresinin ne kadar değişeceğini söyleyen karar kısmı ise sonuç (consequent) değişkenini oluşturmaktadır. Bir araya gelen kuralların verdikleri kararlar bulanık mantık yapısında ele alınmış ve çıktı olarak durulaştırma işleminden geçirildikten sonra tek bir karar üretilmiştir.

Kavşakta gerçekleşebilecek her bir duruma en uygun kararı bulmak zor bir iştir. Çünkü geliştirilen modelde 3^8 adet girdi için "en mantıklı kararların" belirlenmesi gerekmektedir. Bunun yerine karmaşık çok boyutlu arama uzayında en uygun kural ve kararlarının bulunmasını sağlayan bir kural bloğu oluşturmak ve bloğun

verdiği toplam kararı kullanmak yerinde olacaktır. Bloktaki en iyi kural ve kararların bulunmasında Genetik Algoritma kullanılmıştır.

Belirli sayıda bir araya getirilen kurallar gider değeri en az oluncaya kadar eğitim verileri ile optimize edilmiştir. Bir başka değişle katmanın verdiği son kararın en iyi karar olması sağlanıncaya kadar, katmandaki kurallar Genetik Algoritma kullanılarak değiştirilmiştir.

Tek bir katman kavşağın en iyi performansta çalışması için yeterli değildir. Bunun yerine yeni katmanlar yaratılmış ve bir önceki katman ile kaynaştırılarak yeni bir kural katmanı oluşturulmuştur. Böylelikle karmaşık çözüm uzayında en iyi çözüme biraz daha yaklaşılmıştır.

Yeni katmanlar oluşturulurken kural bloğunun döngü sonundaki kararı ile bir önceki katmanının kararı birbiri ile kaynaştırılmıştır. Kaynaştırma işlemi için gereken performans değeri sıradan kavşaktan ayrılan araç miktarı ile kural tabanlı çalışan kavşaktan ayrılan araç miktarının karşılaştırılması ile bulunmuştur. İleriki çalışmalarda performans hesabı yapılırken sıradan kavşak yerine kural tabanlı kavşağın bir önceki performans değerleri dikkate alınarak bir çözüm geliştirilebilir. Böylelikle sıradan kavşağın yeterli olmadığı durumlarda önerilen çözüm yerine, kural tabanlı kavşağın yeterli omadığı durumlarda bir çözüm önerisi geliştirilmiş olunur.

Bu çalışmada bir kavşağın uyarlamalı çalışması sağlanarak bir şehrin trafik sorununa çözüm bulmaya çalışılmıştır. Yerel bir çözüm sağlayan uyarlamalı kavşaklar evrensel bir çözüm yaratmakta kullanılacak en önemli aracı unsurdurlar. Tek başlarına en uygun çözümü bulan kavşaklar ileride merkezi bir sunucudan aldıkları bilgiler ile bir kaç döngü sonrasında nasıl bir davranış sergilemeleri gerektiğine karar verebileceklerdir.

KAYNAKLAR

[1] He, J., Hou, Z., Ant colony algorithm for traffic signal timing optimization, Advances in Engineering Software 43 pp.14–18, **2012**

[2] Hunt, P.B., Robertson, D.L., Bretherton, R.D., Royle, M.C.. The SCOOT on-line traffic signal optimisation technique. Traffic Engineering & Control 23, pp.190–199,**1982**

[3] Ataslar, Iftar, A., Ulasim ağlari için dış merkezli yönlendirme kontrolu algoritması, Elektrik-Elektronik-Bilgisayar Mühendisliği 7. Ulusal Kongresi Bildirileri, ss. 380-383, Ankara, Eylül, **1997**

[4] J.G Bender “An overvıew system studıes of automated highway system” IEEE Transactions on Vehicular Technology, c, VT-40 s. 82-99, **1991**

[5] Dion F., Hellinga B., A rule-based real-time traffic responsive signal control system with transit priority: application to an isolated intersection, Transportation Research Part B 36 pp.325–343, **2002**

[6] Novák, V., Perfilieva, I. and Močkoř, J. Mathematical principles of fuzzy logic Dodrecht: Kluwer Academic. ISBN 0-7923-8595-0, **1999**

[7] Zadeh, L.A. (1965). “Fuzzy sets”,Information and Control (3): 338–353.

[8] Beasley, D., Bull, D.R., and Martın, R.R., a. An Overview of Genetic Algorithms: Part 1, Fundamentals. University Computing, Vol.15(2), pp.58–69, UK, **1993**

[9] Mitchell, Melanie An Introduction to Genetic Algorithms. Cambridge, MA: MIT Press. ISBN 9780585030944, **1996**

[10] Whitley, Darrell. "A genetic algorithm tutorial". Statistics and Computing 4 (2): pp.65–85. doi:10.1007/BF00175354, **1994**

[11] Banzhaf, Wolfgang, Nordin, Peter; Keller, Robert; Francone, Frank. Genetic Programming – An Introduction. San Francisco, CA: Morgan Kaufmann. ISBN 978-1558605107, **1998**

[12] Beldek, U., Leblebicioglu K., A new systematic and flexible method for developing hierarchical decision-making models. **2013**

[13] Hunt, P.B., Robertson, D.L., Bretherton, R.D., Royle, M.C.,. The SCOOT on-line traffic signal optimisation technique. Traffic Engineering & Control 23, 190–199. **1982**

[14] Robertson, D.I.,. Research on the TRANSYT and SCOOT method of signal coordination. ITE Journal, pp. 36–40, **1986**

[15] Li Ruimin. Urban road traffic management. Beijing: People's Traffic Press; **2009**

[16] Yang Jindong, Yang Dongyuan. Optimized signal time model in signaled intersection. J Tongji Univ;29(7):789–94, **2001**

[17] Motawej F., Bouyekhf R., Abdellah El Moudni, A dissipativity-based approach to traffic signal control for an over-saturated intersection Journal of the Franklin Institute 348 (2011) 703–717, **2011**

[18] Novák, V. "Are fuzzy sets a reasonable tool for modeling vague phenomena?", Fuzzy Sets and Systems 156 (2005) 341—348

[19] Jain, L.C. and Martin, N.M., Fusion of Neural Networks, Fuzzy Sets, and Genetic Algorithms: Industrial Applications, 1st Edn., CRC Press, FL, USAISBN: 0849398045, **1998**.

[20] Defuzzification: criteria and classification, from the journal Fuzzy Sets and Systems, Van Leekwijck and Kerre, Vol. 108 , pp. 159-178, **1999**

[21] Cover, Thomas M.; Thomas, Joy A. Elements of Information Theory. John Wiley and Sons. p. 254, **2006**

[22] Diakaki C., Papageorgiou M., Aboudolas K., A multivariable regulator approach to traffic-responsive network-wide signal control, Control Engineering Practice 10 183–195, **2002**

EK - 1

1 - Algoritmanın yazınsal gösterimi

---

1. Başlangıçta M adet kuraldan oluşan bir set (kromozom) çözüm uzayından seçilir.
2. Oluşturulan set in her bir kuralı için “en uygun karar” ataması eller yapılır.
3. Kromozomun ***C*** adet döngü zamanı sonundaki $\boldsymbol{Cost_{first}}$ değeri hesaplanır.
4. Her bir döngü zamanı sonunda standart kavşağa göre olan perfromans değeri hesaplanır.
5. $\boldsymbol{Cost_{new} = Cost_{first}}$
6. $\boldsymbol{Cost_{old} = Cost_{first}}$
7. **N =** $\boldsymbol{N_{initial}}$
8. $\boldsymbol{b = b_{initial}}$
9. **iterasyonSayısı = 0**
10. Kural havuzundan rastgele N adet kural seti bir başka değişle N adet kromozom seçilir.
11. $\boldsymbol{Cost_{new} > Cost_{old}}$
    a. Yanlış ise
        i. N > **MIN_KROMOZOM_SAYISI**
            1. Doğru ise
                a. iterasyonSayısı <**MAX_ ITERASYON_SAYISI**
                    i. doğru ise

1. Her bir kromozom bir üst katmandaki kromozom ile füzyon edilip ***C*** adet döngü zamanı sonunda ***Cost*** değerleri hesaplanır.
2. Her bir kromozomun ***fitness*** değeri hesaplanır.
3. En iyi fitness değerine ($\boldsymbol{Cost_{best}}$) sahip olan kromozom alınıp bir sonraki iterasyonda kullanılmak üzere yeni oluşturulacak populasyona atılır.
4. Her bir kormozomun her bir döngü döngü zamanı sonunda standart kavşağa göre olan ***perfromans*** değeri hesaplanır.
5. $\boldsymbol{Cost_{new} = Cost_{best}}$
6. En kötü ***fitness*** değerine sahip olan kromozom ise populasyondan atılır.
7. En kötü kromozom hariç diğerler kromozomlar ***crossover*** ve ***mutasyon*** işlemlerinden geçirilip yeni bir generasyon üretilir.
8. iterasyonSayısı = iterasyonSayısı + 1
9. Git madde 11

ii. Yanlış ise

1. N = N – 1
2. iterasyonSayısı = 0
3. Git madde 10.

2. Yanlış ise

a. totalIterasyon < **MAX_TOTAL_ITERATION_SAYISI**

i. Doğru ise

1. totalIterasyon = totalIterasyon + 1

2. N = $N_{initial}$
3. Performans hesabındaki denge değeri bir miktar artırılır. Böylece eski başarılı kromozomların performans değerinin etkisi artırılmış olur (***b = b + 0.025***)
4. Git madde 10

ii. Yanlış ise

1. $\boldsymbol{Cost_{final} = Cost_{old}}$
2. SON.

b. Doğru ise

i. füzyonSayısı < **MAX_FÜZYON_SAYISI**

1. doğru ise
   a. $\boldsymbol{Cost_{old} = Cost_{new}}$ (birinci füzyon işlemi başarılı olmuştur bir sonraki füzyon işlemine geçilir)
   b. git madde 7
2. yanlış ise
   a. $\boldsymbol{Cost_{final} = Cost_{old}}$
   b. SON

EK – 2

1. Algoritmanın Kaynak Kodu

TSCMaın.m

```
function [] = TSCmain()
    clear;
    clc;
    close all;
    disp('Algorithm Started');

    %Config Param and Definition
    %
    %
    LOGTYPE_TSCMAIN            = 'TSCMainLogs';
    LOGTYPE_TSC                = 'TSCLogs';
    LOGTYPE_TSCPERVAL          = 'TSCPerValLogs';
    INPUT_FLOW_RATE_FOLD_NAME   =
'E:\PROJECTS\ADVANCE_TRAFFIC_CONTROL_SYSTEM\AKADEMIK\opt\DERS13_Rev_01_03\INPUTS\inFlwRates.mat';
    FIS_STRUCT_FOLD_NAME       =
'E:\PROJECTS\ADVANCE_TRAFFIC_CONTROL_SYSTEM\AKADEMIK\opt\DERS13_Rev_01_03\testVar\tscAnfis6.mat';
    %'E:\PROJECTS\ADVANCE_TRAFFIC_CONTROL_SYSTEM\AKADEMIK\opt\DERS13_Rev_01_02\INPUTS\inFlwRates.mat'
    %'E:\PROJECTS\ADVANCE_TRAFFIC_CONTROL_SYSTEM\AKADEMIK\opt\DERS13\testVar\inFlwRates2.mat'
```

```
LOGLEVEL_INFO              = 0;
LOGLEVEL_WARNING           = 1;
LOGLEVEL_ERROR             = 2;
LOGLEVEL_FATAL_ERROR       = 3;

CHANGE_PERF_VALUE_THEN_FUSION   = 0;
DECREASE_RULE_COUNT_THEN_FUSION = 1;
%-------------------------------------------

%Explanation
%write clearly what is your code doing
ALGORTIHM_EXPLANATION = ...
[%write your comments to below
'-----------------------------------------------------------------\n'...
'                ALGORTIHM EXPLANATION                          \n'...
'  Adjust the old performace according to the new rules cost    \n'...
'Improve the effectiveness of old rules or decrese accordign to \n'...
'new fusioned rules cost.                                       \n'...
'-----------------------------------------------------------------\n'...
'                   PARAMATERS                                  \n'
];

str = sprintf(ALGORTIHM_EXPLANATION);
%-------------------------------------------

%Paramaters
CYCLE_COUNT                = 200;  %how many times a cycle will run
```

```
CYCLE_LENGTH               = 140;  %1 cycle length in second
GA_INPUT_COUNT              = 18;   %Genetic algorithm input count
FUSION_RULE_COUNT           = 8;    %new rules count. First is 20 rule.
FUSION_MAX_RETRY_COUNT      = 8;    %max retry count. finish, if overreached
FUSION_MIN_RULE_COUNT       = 2;    %
FUSION_LEVEL                = 4;
PERFORMANCE_BALANCE_VALUE   = 0.5;
PARENT_NUMBER               = 50;
GENETIC_ALGORITHM_ITERATION = 100;
%-----------------------------------------------

%global parameters
global cycleLength;
global OnePhase2;
global logFolderPath;
global inFlwRts;

%get initial Parent for 20 rule
initialParent = getInitialParent();

%initial cycle length
cycleLength = CYCLE_LENGTH ;

%initial Fis Rules
fis_struct = load(FIS_STRUCT_FOLD_NAME);
OnePhase2 = fis_struct.OnePhase2;
```

```
%creating Folder
resultFolder = CreateFolder();

%create Log Folder
logFolderPath = CreateLogPath(resultFolder);

%get Input Flowe rates
inFlwRts = getInputFlowRates2(INPUT_FLOW_RATE_FOLD_NAME);
%-----------------------------------------------


%Create and Save Configuration To Config File
configFileName = strcat(resultFolder,'\','config.txt');
fileID = fopen(configFileName,'w+');
fprintf(fileID,'%s',str);
fprintf(fileID,'CYCLE_COUNT = %d \n',CYCLE_COUNT);
fprintf(fileID,'CYCLE_LENGTH = %d \n',CYCLE_LENGTH);
fprintf(fileID,'GA_INPUT_COUNT  = %d \n',GA_INPUT_COUNT);
fprintf(fileID,'FUSION_RULE_COUNT = %d \n',FUSION_RULE_COUNT);
fprintf(fileID,'FUSION_MAX_RETRY_COUNT = %d \n',FUSION_MAX_RETRY_COUNT);
fprintf(fileID,'FUSION_MIN_RULE_COUNT = %d \n',FUSION_MIN_RULE_COUNT);
fprintf(fileID,'FUSION_LEVEL = %d \n',FUSION_LEVEL);
fprintf(fileID,'PERFORMANCE_BALANCE_VALUE  = %2.1f \n',PERFORMANCE_BALANCE_VALUE );
fprintf(fileID,'PARENT_NUMBER = %d \n',PARENT_NUMBER);
fprintf(fileID,'GENETIC_ALGORITHM_ITERATION = %d \n',GENETIC_ALGORITHM_ITERATION);

fprintf(fileID,'-------------------------------------------------------------\n');
```

```
fclose(fileID);
%-----------------------------------------------



%run ordinary intersection and get costOrd adn Intersection behavior
ordIntResultSt = struct('Cost',0,'Intersection',0);
[costOrd interOrd] = TSCord(CYCLE_COUNT);
ordIntResultSt.Cost = costOrd;
ordIntResultSt.Intersection = interOrd;
%-----------------------------------------------


oldParent = zeros(CYCLE_COUNT,GA_INPUT_COUNT);

lowLevelResults(1:1:FUSION_LEVEL + 2) = struct('parent',0,'fisStruct',0,'perform',0,'perfBalanceVal',0);

%initialize.
% first low level dummy case.
for i=1:1:FUSION_LEVEL + 2
   lowLevelResults(i).parent = oldParent;
   lowLevelResults(i).perform = 0;
   lowLevelResults(i).perfBalanceVal = PERFORMANCE_BALANCE_VALUE;
end
%-----------------------------------------------

%    lowLevelResults(1) = struct('parent',0,'fisStruct',0,'perform',0,'perfBalanceVal',0);
```

```
%     lowLevelResults(1).parent = oldParent;
%     lowLevelResults(1).perform = 0;
%     lowLevelResults(1).perfBalanceVal = PERFORMANCE_BALANCE_VALUE;



    %run the first 20 rules
    SaveLog('---------------FIRST RULES STARTED TO IMPLEMENT--------------------',LOGTYPE_TSCMAIN,LOGLEVEL_INFO);


    [finalParent finalPerform bestFisStruct] =
runGA(lowLevelResults,initialParent,ordIntResultSt,PERFORMANCE_BALANCE_VALUE,PARENT_NUMBER,GENETIC_ALGORI
THM_ITERATION,CYCLE_COUNT ,2);
    lowLevelResults(2).parent          = finalParent;
    lowLevelResults(2).perform         = finalPerform;
    lowLevelResults(2).fisStruct       = bestFisStruct;
    lowLevelResults(2).perfBalanceVal   = PERFORMANCE_BALANCE_VALUE;

    str = sprintf('lowLevelResults_%d',2);
    saveStruct(lowLevelResults(2),str,resultFolder);
    %-----------------------------------------------



    %variable initialize
    fusionRuleCount        = FUSION_RULE_COUNT;
    balanceValue           = PERFORMANCE_BALANCE_VALUE;
    retry                  = 0;
```

```
   oldBestEffortParent     = 0;
   level                   = 3;
   IsOldPerfValueChanged   = 0;
   status                  = CHANGE_PERF_VALUE_THEN_FUSION;
   %-------------------------------------------------


   SaveLog('----------NEXT LEVEL-----------', LOGTYPE_TSCMAIN, LOGLEVEL_INFO);

   while (level ~= (FUSION_LEVEL + 3))

      if IsOldPerfValueChanged  == 1
         fprintf('Level:%d executing..., Last best Effort Rules are Selected Again. FusionLueCount:%d \n', level-1, fusionRuleCount);
         str = sprintf('Level:%d executing..., Last best Effort Rules are Selected Again. FusionLueCount:%d', level-1,
fusionRuleCount);
         SaveLog(str, LOGTYPE_TSCMAIN, LOGLEVEL_INFO);
         [finalParent finalPerform bestFisStruct] = runGA(lowLevelResults,oldBestEffortParent,
ordIntResultSt,balanceValue,PARENT_NUMBER,GENETIC_ALGORITHM_ITERATION,CYCLE_COUNT,level);
      else
         fprintf('Level:%d executing..., Random new %d rules selected \n', level-1, fusionRuleCount);
         str = sprintf('Level:%d executing..., random new %d rules selected', level-1, fusionRuleCount);
         SaveLog(str, LOGTYPE_TSCMAIN, LOGLEVEL_INFO);
         [finalParent finalPerform bestFisStruct] = runGA(lowLevelResults,
generateRandSmallParent(fusionRuleCount,size(initialParent,2),50) ,ordIntResultSt,balanceValue,PARENT_NUMBER,
GENETIC_ALGORITHM_ITERATION, CYCLE_COUNT,level);
      end
```

```
switch (IsResultBetterThanOld(lowLevelResults,finalPerform,level))

    case 1
        % if the new result is better than old result
        % fusion the new rules
        disp('OK >> NEW RULES ARE BETTER THAN OLD RULES');
        SaveLog('OK >> NEW RULES ARE BETTER THAN OLD RULES', LOGTYPE_TSCMAIN, LOGLEVEL_INFO);

        lowLevelResults(level).parent         = finalParent;
        lowLevelResults(level).perform        = finalPerform;
        lowLevelResults(level).fisStruct      = bestFisStruct;
        lowLevelResults(level).perfBalanceVal = balanceValue;

        str = sprintf('lowLevelResults_%d',level);
        %saveStruct(lowLevelResults(2:1:end),str,resultFolder);
        saveStruct(lowLevelResults(2:1:level),str,resultFolder);


        fusionRuleCount                 = FUSION_RULE_COUNT;
        status                          = CHANGE_PERF_VALUE_THEN_FUSION;
        retry                           = 0;
        level                           = level + 1;
        IsOldPerfValueChanged           = 0;
        %-----------------------------------------------
```

```
        if level < (FUSION_LEVEL + 3)
            disp('----------NEXT LEVEL is Implementing-----------');
            SaveLog('----------NEXT LEVEL is Implementing-----------', LOGTYPE_TSCMAIN, LOGLEVEL_INFO);
        end




    case 0
        % if new result is not different than the old result
        % increse the new result power.
        SaveLog('NEW RULES ARE NOT GOOD ENOUGH', LOGTYPE_TSCMAIN, LOGLEVEL_WARNING);
        disp('NEW RULES ARE NOT GOOD ENOUGH');

        switch status
            case CHANGE_PERF_VALUE_THEN_FUSION
                IsOldPerfValueChanged = 1;
                oldBestEffortParent = finalParent;
                LastBestEffortPerformValue = lowLevelResults(level-1).perform.perValue ;
                [newPerValueForOldRules] = CalculateNewPerfValueOfOldRules(lowLevelResults,finalPerform,level);
                lowLevelResults(level-1).perform.perValue =  lowLevelResults(level-1).perform.perValue +
newPerValueForOldRules;
                status = DECREASE_RULE_COUNT_THEN_FUSION;
                disp('Performance Value Changed. Rule Count not decreased');
                SaveLog('Performance Value Changed. Rule Count not decreased', LOGTYPE_TSCMAIN,
LOGLEVEL_WARNING);
            case DECREASE_RULE_COUNT_THEN_FUSION
                IsOldPerfValueChanged = 0;
```

```
                lowLevelResults(level-1).perform.perValue = LastBestEffortPerformValue;
                fusionRuleCount = fusionRuleCount - 1;
                status = CHANGE_PERF_VALUE_THEN_FUSION;
                retry = retry + 1;
                 disp('Rule Count Decreased. Perf Value same as before.');
                SaveLog('Rule Count Decreased. Perf Value same as before.', LOGTYPE_TSCMAIN, LOGLEVEL_WARNING);
            end

        end

         if (FUSION_MAX_RETRY_COUNT <= retry)
            disp('OPTIMIZATION WAS STOPPED');
            SaveLog('OPTIMIZATION WAS STOPPED. NEXT fusion will not implement...', LOGTYPE_TSCMAIN,
LOGLEVEL_ERROR);
            str = sprintf('Global Minimum Cannot be found at level:%d',level-1);
            SaveLog(str, LOGTYPE_TSCMAIN, LOGLEVEL_ERROR);
            str = sprintf('Level:%d discarded.',level-1);
            SaveLog(str, LOGTYPE_TSCMAIN, LOGLEVEL_ERROR);
            break;
         end

    end
    %-----------------------------------------------

    %Plot and save results
    plotOrdIntQueue(lowLevelResults, CYCLE_COUNT);
    saveStruct(lowLevelResults(2:1:FUSION_LEVEL+2),'lowLevelResults',resultFolder);
    plotInputFlwRt(resultFolder);
```

```
    plotTheFigures(lowLevelResults,resultFolder,CYCLE_COUNT);
    %-----------------------------------------------

end

function [newPerfValue]  = CalculateNewPerfValueOfOldRules(lowLevelResults,finalPerform,level)
    diff = finalPerform.costRulInter - lowLevelResults(level-1).perform.costRulInter;

    signumDiff = sign(diff);

    newPerfValue = signumDiff * 0.05;

end

function [res] = IsResultBetterThanOld(lowLevelResults,finalPerform, level)

    global cycleLength;

    diff = lowLevelResults(level-1).perform.costRulInter(cycleLength) - finalPerform.costRulInter(cycleLength);

    if(diff > 0)
        res = 1;
    else
        res = 0;
    end

end
```

```
function [parent ] = generateRandSmallParent(row,col,changeBitNum)

    parent(1:1:row,1:1:col) = 0;

    for bit=1:1:changeBitNum
        rw = randi(row,1);
        co = randi(col,1);
        if(parent(rw,co) == 0)
            parent(rw,co) = 1;
        end
    end
end


function [] = plotInputFlwRt(folder)

    global inFlwRts;

    h = figure;
    hold on;
    grid;

    subplot(4,1,1), plot(inFlwRts(1,:), '-b');
    axis([0 2800 0 5]);
```

```
    title('Input Flow Rates');
    ylabel('inFlwRt1');

    subplot(4,1,2), plot(inFlwRts(2,:), '-r');
    axis([0 2800 0 5]);
    title('Input Flow Rates');
    ylabel('inFlwRt2');

    subplot(4,1,3), plot(inFlwRts(3,:), '-g');
    axis([0 2800 0 5]);
    ylabel('inFlwRt3');
    title('Input Flow Rates');

    subplot(4,1,4), plot(inFlwRts(4,:), '-c');
    title('Input Flow Rates');
    axis([0 2800 0 5]);
    ylabel('inFlwRt4');
    xlabel('Cycle');
    hold off;

    figName = strcat(folder,'\','InputFlowRate');
    saveas(h,figName,'fig');

end

function [] = plotOrdIntQueue(performRes,cycleCount)
        figure;
```

```
        grid on
        for i =1:1:cycleCount+1
            queue(i,:) = performRes(2).perform.ordinaryIntersec(i).queue(:,end)';
        end
        bar(queue);
        axis([0 cycleCount+2 0 600]);
        xlabel('Cycle');
        ylabel('Queue');
        lh = legend('line1','line2','line3','line4');
        set(lh, 'Location', 'NorthWest');
        title('Standart Intersection Queue Lenghts');
end

function [res] = getInputFlowRates2(inPutFlowRateFolderName)

    inputs = load(inPutFlowRateFolderName);
    inFlwRts = inputs.inFlwRts;
    col = size(inFlwRts(1,:),2);

    for indx=1:1:col

        if(inFlwRts(1,indx) < 0)
            inFlwRts(1,indx) = 0;
        end
        if(inFlwRts(1,indx) > 4)
            inFlwRts(1,indx) = 4;
        end
```

```
        if(inFlwRts(2,indx) < 0)
            inFlwRts(2,indx)= 0;
        end
        if(inFlwRts(2,indx) > 4)
            inFlwRts(2,indx) = 4;
        end

        if(inFlwRts(3,indx) < 0)
            inFlwRts(3,indx) = 0;
        end
        if(inFlwRts(3,indx) > 4)
            inFlwRts(3,indx) = 4;
        end

        if(inFlwRts(4,indx) < 0)
            inFlwRts(4,indx)= 0;
        end
        if(inFlwRts(4,indx)> 4)
            inFlwRts(4,indx) = 4;
        end

    end

    res = [inFlwRts(1,:);inFlwRts(2,:);inFlwRts(3,:);inFlwRts(4,:)];
end
```

```
function [res] = saveStruct(str,strName,folderName)

   structName = strcat(folderName,'\',strName,'.mat');

   saveableStr = struct('savedStructure',0);

   saveableStr.savedStructure = str;

   save(structName,'-struct','saveableStr');

end

function [res] = getInitialParent()


res =[
        0 0 0 1 0 1 0 1   0 0 0 0 0 0 0 0    1 1
        0 0 1 1 0 1 0 1   0 0 0 0 0 0 0 0    0 1
        1 0 1 0 0 1 0 0   0 0 0 0 0 1 0 0    1 0
        0 1 1 0 0 1 0 1   0 0 0 0 0 0 0 0    1 0
        0 0 0 1 1 0 0 1   0 0 0 0 0 0 0 0    1 1
        1 0 0 1 1 0 0 1   1 0 0 1 1 0 0 1    0 0
        0 0 0 0 0 0 1 1   0 0 0 0 0 0 0 0    0 1
        0 0 0 0 0 1 1 0   0 0 0 0 0 1 1 1    0 1
```

```
        0 0 0 0 0 1 1 1   0 0 0 0 1 0 1 1   0 1
        0 0 0 0 1 0 1 1   0 0 0 0 1 0 1 1   0 1
        0 0 0 0 1 1 1 1   0 0 0 0 1 0 1 1   1 1
        0 0 1 1 0 0 0 0   0 0 1 1 0 1 0 1   0 0
        0 0 1 1 0 0 0 0   0 1 1 1 0 1 0 1   0 0
        0 0 1 1 0 0 0 0   1 0 1 1 0 1 0 1   1 1
        1 0 1 1 0 0 0 1   1 1 1 1 0 1 0 1   0 0
        1 1 1 1 1 1 0 1   0 0 0 1 0 1 0 0   0 0
        1 1 1 0 0 0 0 0   0 1 0 1 0 0 0 0   1 0
        1 1 0 0 1 1 0 0   1 0 0 1 0 1 1 0   1 1
        1 0 1 0 1 0 0 0   0 0 0 0 1 0 1 0   0 0
        0 0 1 1 1 0 0 1   0 0 0 0 1 1 0 1   0 0
    ];
end
```

```
function [finalBestParent finalPerformance bestFisStruct] = runGA(lowLevelRes,initialParent, ordIntResSt,balanceValue
,PARENT_NUMBER,GENETIC_ALGORITHM_ITERATION, cycleCount, fusionLevel)
    % inFlwRt x 4=
    %     00 - ZERO
    %     01 = SMALL
    %     10 - MEDIUM
    %     11 - BIG

    % RtChInFlwRt x 4=
    %     00 - NEGATIVE BIG
    %     01 = NEGATIVE
    %     10 - POSITIVE
    %     11 - POSITIVE BIG

    % initialQueue x 4=
    %     00 - ZERO
    %     01 = SMALL
```

```
%     10 - MEDIUM
%     11 - BIG

% effect Green Time Change =
%     000 - NEGATIVE BIG
%     001 = NEGATIVE MEDIUM
%     010 - ZERO
%     011 - POSITIVE MEDIUM
%     100 - POSITIVE BIG

%lowLevelResults(1) = struct('parent',0,'fisStruct',0,'perform',0);
parentNumber = PARENT_NUMBER;

perfRes(1:1:parentNumber) =  ...
   struct(                          ...
         'perValue',                  0, ...
         'fitness',                  0, ...
         'costRulInter',              0, ...
         'costOrdInter',              0, ...
```

```
        'effGrPh2RuleInter',            0, ...
        'effGrPh2OrdInter',             0, ...
        'bestEffortRuleBaseIntersec',   0, ...
        'ordinaryIntersec',             0 ...
       );


Populations(1:GENETIC_ALGORITHM_ITERATION) = InitialPopulationGenerator(initialParent,parentNumber);

%create FisStruct
levelNumber = size(lowLevelRes,2);
for lv=1:1:levelNumber
   rw = size(lowLevelRes(lv).parent,1);
   lowLevelRes(lv).fisStruct = CreateFisRules(lowLevelRes(lv).parent, rw);
end


for number=1:1:GENETIC_ALGORITHM_ITERATION
```

```
    for parNum=1:1:parentNumber
        row                 = size(Populations(number).pop(parNum).parent,1);
        curFisStruct        = CreateFisRules(Populations(number).pop(parNum).parent, row);

%           str = sprintf('%d ',Populations(number).pop(parNum).parent);
%           SaveLog(str, 'GA_Rule', 0);
%           SaveLog('-----------------------------------------', 'GaRule', 0);

        %[perfRes(parNum)]  = calculateEffGrTimeAndPerf(lowLevelRes,curFisStruct,nextLevel,inFlwRts);

        [perfRes(parNum)]   = getPerformance(lowLevelRes,curFisStruct,ordIntResSt,balanceValue,cycleCount,fusionLevel);
        Populations(number).pop(parNum).perf = perfRes(parNum);
    end

    str = sprintf('%d. iteration executed',number);
    SaveLog(str, 'runGA', 0);

    [Populations(number)]   = CalculateFitness(Populations(number));
```

```
        [SortedPops FIT IX]     = SortTheParntsAccordingTheFitness(Populations,number,parentNumber);

        Populations(number + 1) = GenerateNewPopulation(SortedPops(1));
    end

    %plotTheFigures(Populations(end-1).pop(1).perf);
    SaveLog('itt finished get time - 1', 'runGA', 0);
    finalBestParent  = SortedPops(1).pop(1).parent;
    finalPerformance = SortedPops(1).pop(1).perf;
    row              = size(finalBestParent,1);
    SaveLog('itt finished  get time - 2', 'runGA', 0);
    bestFisStruct    = CreateFisRules(finalBestParent , row);
    SaveLog('itt finished  get time - 3', 'runGA', 0);
end

function [population] = CalculateFitness(population)

    perNumber = size(population.pop,2);
```

```
    for per=1:1:perNumber
        % find some way to calculate the fitness
        population.pop(per).perf.fitness = 1 / population.pop(per).perf.costRulInter(end);
    end

end

function [population] = GenerateNewPopulation(SortedPops)

    %[SortedPops FIT IX] = SortTheParntsAccordingTheFitness(Populations,parentNumber);
    % get two the best performance - first two of them is best
    % crossover other s
    population          = CrossoverTheParents(SortedPops);
    % add mutation
end

function [SortedPops sortedFitness IX] = SortTheParntsAccordingTheFitness(Populations,popClusterCount,parentNumber)

    SortedPops = Populations;
```

```
parentCountInEachPopCluster = parentNumber;
%pop   = size(Populations,2);

totalParent  = popClusterCount * parentCountInEachPopCluster;

Fit = 0;
fitness(1:1:parentCountInEachPopCluster) = 0;

for popClCount = 1:1:popClusterCount
  for par=1:1:parentCountInEachPopCluster
     fitness(par) = Populations(popClCount).pop(par).perf.fitness;
  end
  Fit = [Fit fitness];
end

Fit = Fit(2:1:end);

[sortedFitness, IX] = sort(Fit, 2 , 'descend');
```

```
    %disp(sortedFitness);
    for indx=1:1:totalParent
     clust      = fix((IX(indx) -1) / parentCountInEachPopCluster ) + 1;
     subIndx    = IX(indx) - (clust-1) * parentCountInEachPopCluster;

     newClust    = fix((indx-1)/parentCountInEachPopCluster) + 1;
     newSubIndx  = indx - (newClust - 1) * parentCountInEachPopCluster;

     SortedPops(newClust).pop(newSubIndx) = Populations(clust).pop(subIndx);

    end

end

function [newPopulation] = CrossoverTheParents(population)

    newPopulation = population;
    parNum = size(population.pop,2);
```

```
%*************************ELITIZM*********************************%
% first two of the parents are the best put them in front
newPopulation.pop(1) = population.pop(1);
newPopulation.pop(2) = population.pop(2);
parent1 = population.pop(1).parent;
parent2 = population.pop(2).parent;


%best chromozome crossover and added to the end
[Offspring1 Offspring2] = Cross(parent1, parent2);
newPopulation.pop(end).parent   =  Offspring1;
newPopulation.pop(end-1).parent =  Offspring2;
%*****************************************************************%

tempArray = (3:1:parNum-2);
col = size(tempArray,2);
```

```
    %*****************Select Randon from ordered 20 population**********************
    for p=3:2:parNum-2

        parent1 = population.pop(p).parent;
        parent2 = population.pop(p+1).parent;

        [Offspring1 Offspring2] = Cross(parent1, parent2);

        newPopulation.pop(p).parent = Offspring1;
        newPopulation.pop(p+1).parent = Offspring2;

    end
    %*************************************************************************
    % first two of the parents are the best put them in front
end

function [offspring1 offspring2] = Cross(par1, par2)
    [row col] = size(par1);
```

```
    offspring1((1:1:row),(1:1:col)) = 0;
    offspring2((1:1:row),(1:1:col)) = 0;

    for r=1:1:row
        crosoverPoint = randi(col,1);
        offspring1(r,:) = [par1(r,(1:1:crosoverPoint)) par2(r,(crosoverPoint+1 :1:end))];
        offspring2(r,:) = [par2(r,(1:1:crosoverPoint)) par1(r,(crosoverPoint+1 :1:end))];
    end

    offspring1 = AddMutation(offspring1, 10);
    offspring2 = AddMutation(offspring2, 10);

end

function [Populations] = InitialPopulationGenerator(InitParent,parentNumber)

    ParentsSt = struct('parent',0,'perf',0);

    ParentsSt(1).parent = InitParent;
```

```
    ParentsSt(1).perf   = 0;

    %ParentsSt(2).parent = InitParent;
    %ParentsSt(2).perf   = 0;

    % generate 20 different Parent
    for i =1:1:parentNumber-1
        ParentsSt(i+1) = GenerateNewParent(ParentsSt(i).parent);
    end

    Populations = struct('pop',0);
    Populations.pop = ParentsSt;

end

function [NewParentSt] = GenerateNewParent(parent)
    NewParentSt = struct('parent',0,'perf',0);
    NewParentSt.parent = AddMutation(parent,10);
end
```

```
function [newParent] = AddMutation(parent, per)
    [row col] = size(parent);
    mutationNumber = round((col/100) * per);
    newParent = parent;
    for r=1:1:row
        for c = 1:1:mutationNumber
            indx = randi(col,1);
            if(parent(r,indx)==1)
                newParent(r,indx) = 0;
            else
                newParent(r,indx) = 1;
            end
        end
    end
end
```

```
function [result] = getPerformance(lowLevelRes,curFisStruct,ordIntResSt,balanceValue,cycleCount,fusionLevel)
    SaveLog('get time - 0  ', 'getPerformance', 0);
    N=cycleCount;
    effGrPh2 = 70;
    totalQueuLen(:,1) = [0;0;0;0];%this value not used
    effGrPhOrdInt = zeros(1,N);
    effGrPhRulInt = zeros(1,N);

    intersection(1:1:N+1) = ...
    struct(                                 ...
          'queue',            zeros(4,N),    ...
          'totalQueue',       zeros(4,N+1),  ...
          'effGrPh2',         70,            ...
          'leavedCarNum',     zeros(4,N+1),  ...
          'totalLeavedCar',   0 ,            ...
          'totInFlwRt',       [0;0;0;0]      ...
       );
```

```
result =  ...
struct(                                ...
        'perValue',                    0, ...
        'fitness',                     0, ...
        'costRulInter',                0, ...
        'costOrdInter',                ordIntResSt.Cost, ...
        'effGrPh2RuleInter',           0, ...
        'effGrPh2OrdInter',            0, ...
        'bestEffortRuleBaseIntersec',  intersection, ...
        'ordinaryIntersec',            ordIntResSt.Intersection  ...
    );



SaveLog('get time -1 ', 'getPerformance', 0);
for c=1:1:N

    if c>1
```

```
        %CALCULATING EFFECTIVE GREEN TIME PHASE 2
        [effGrPh2]                = CalculateEffGrTimePh2(lowLevelRes,curFisStruct,intersection,effGrPh2,c,fusionLevel);
    end

    intersection(c).effGrPh2        = effGrPh2;

    %RUN TRAFFIC SIGN CONTROLLER
    intersection(c)                 = runTSC(intersection(c) ,effGrPh2, c);

    %pass the queue to the next cycle
    intersection(c+1).queue(:,1)    = intersection(c).queue(:,end);
    totalQueuLen(:,c)               = intersection(c).queue(:,end);
    intersection(c).totalQueue      = totalQueuLen(:,c);
    intersection(c).totalLeavedCar  = sum(sum(intersection(c).leavedCarNum,2),1);

end
SaveLog('get time - 2 ', 'getPerformance', 0);

%calculate ruleBase Intersection cost
```

```
    intVal          = sum(totalQueuLen,1);
    cost(1:1:N)     = 0;
    cost(1)         = intVal(1);
    for i=2:1:N
        cost(i)     = cost(i-1) + intVal(i);
    end

    result.costRulInter = cost;
    %-----------------------------------------

    %add ordinary intersection result
    %result.costOrdInter     = ordIntResSt.Cost;
    %result.ordinaryIntersec = ordIntResSt.Intersection;
    %-----------------------------------------
     SaveLog('get time - 3 ', 'getPerformance', 0);
    %Calculate the performance
    [perValue]                      =
TSCevalPerVal(intersection,ordIntResSt.Intersection,lowLevelRes,curFisStruct,balanceValue,cycleCount,fusionLevel);
```

```
    result.perValue                    = perValue;
    result.bestEffortRuleBaseIntersec   = intersection;
     SaveLog('get time - 4 ', 'getPerformance', 0);

    for c=1:1:N
        effGrPhOrdInt(c) = ordIntResSt.Intersection(c).effGrPh2;
        effGrPhRulInt(c) = intersection(c).effGrPh2;
    end

    result.effGrPh2RuleInter = effGrPhRulInt;
    result.effGrPh2OrdInter  = effGrPhOrdInt;
    %-----------------------------------------

end
```

```
function [effGrPh2] = CalculateEffGrTimePh2(lowLevelRes,curFisStruct,intersection,effGrPh2,c,fusionLevel)


    %initial flwrtchange
    inputFlwRtCh(:,1) = [0;0;0;0];%this value not used



    if (c==2)
        inputFlwRtCh(:,c) = [100;100;100;100]; %assign initial contion for new begining high input for all direction
    else
        inputFlwRtCh(:,c) = ((intersection(c-1).totInFlwRt - intersection(c-2).totInFlwRt) ./ intersection(c-2).totInFlwRt ) * 100;
    end


    [totInFlowRatF initialQueueF] = filterFISInput(intersection(c-1).totInFlwRt,inputFlwRtCh(:,c),intersection(c).queue(:,1));
```

```
if (fusionLevel == 2)
    actualDecison = evalfis([totInFlowRatF' initialQueueF'],curFisStruct);
else



    tempDecision = zeros(1,fusionLevel);

    %dummy case
    tempDecision(1)  =  evalfis([totInFlowRatF' initialQueueF'],lowLevelRes(1).fisStruct);

    tempDecision(2)  =  evalfis([totInFlowRatF' initialQueueF'],lowLevelRes(2).fisStruct);

    for lv=3:1:fusionLevel
        tempDecision(lv)   =    evalfis([totInFlowRatF' initialQueueF'],lowLevelRes(lv).fisStruct);
        tempDecision(lv)   =    lowLevelRes(lv-1).perform.perValue(c) * tempDecision(lv - 1)  + (1 - lowLevelRes(lv-
1).perform.perValue(c)) * tempDecision(lv);
    end
```

```
        decision         = evalfis([totInFlowRatF' initialQueueF'],curFisStruct);
        actualDecison    = lowLevelRes(fusionLevel-1).perform.perValue(c) * tempDecision(end) + (1-lowLevelRes(fusionLevel-
1).perform.perValue(c)) * decision;

        %disp([tempDecision actualDecison]);
    end
    effGrPh2 = effGrPh2 + effGrPh2 * actualDecison;

    effGrPh2 = min(effGrPh2 , 110);
    effGrPh2 = max(30, effGrPh2);

end
```

ÖZGEÇMİŞ

Kimlik Bilgileri

Adı Soyadı : Zeki Güler

Doğum Yeri: Cizre

Medeni Hali : Evli

E-Posta : zkglr53@gmail.com

Adresi : Gençlik Cad. Kızılcık Sok. 14/5 Anıttepe/ANKARA

Eğitimi

Lise : Çağrıbey Anadolu Lisesi

Lisans : Anadolu Üniversitesi

Yüksek Lisans: : Hacettepe Üniversitesi

Yabancı Dil ve Düzeyi

İnglizce : Çok iyi

İş Deneyimi

3 yıl Ortana Firmasında ArGe mühendisi

Deneyim Alanı

Kontrol Kuramı, Bulanık Mantık Yapıları, Akıllı Sistemler, Sinyal İşleme

Tezden Üretilmiş Tebliğ ve/veya Poster Sunumu İle Katıldığı Toplantılar

Otomatik Kontrol Ulusal Toplantısı (TOK) 2013

.