

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

**ARAÇ YÖRÜNGELERİ İÇİN AYRIK OLAY SİSTEM
UYGULAMALARI**

142 640

YÜKSEK LİSANS TEZİ

**Elektrik Müh. Murat GÜZEL
(504001261)**

142640

**Tezin Enstitüye Verildiği Tarih : 5 Aralık 2003
Tezin Savunulduğu Tarih : 14 Ocak 2004**

Tez Danışmanı : Doç.Dr. Salman KURTULAN
Diğer Jüri Üyeleri : Yrd. Doç.Dr. Neslihan Serap ŞENGÖR
Prof.Dr. Leyla GÖREN

ARALIK 2003

**T.C. YÜKSEKÖĞRETİM
BİLİMLERİ ENSTİTÜSÜ
BİLİMLERİN İNTEGRASYON MERKEZİ**

ÖNSÖZ

Araç Yörüngelerinde Ayrık Olay Sistem Uygulamalarının araştırılması olan bu çalışmada yardımlarını esirgemeyen sayın danışmanım Doç.Dr.Salman Kurtulan'a ve sayın hocam Yrd.Doç.Dr. Neslihan Serap Şengör'e teşekkür ederim.

Murat Güzel

Aralık 2003



İÇİNDEKİLER

KISALTMALAR	v
TABLO LİSTESİ	vi
ŞEKİL LİSTESİ	vii
SEMBOL LİSTESİ	viii
ÖZET	ix
SUMMARY	x
1. GİRİŞ	1
1.1. Giriş ve Çalışmanın Amacı	1
2. AYRIK OLAYLI SİSTEMLERE GİRİŞ	3
2.1. Olay Kavramı	3
2.1.1. Zaman Yürütümlü Sistem ile Olay Yürütümlü Sistemin Karşılaştırılması	3
2.2. Ayrık Olaylı Sistemleri	4
3. PETRİ AĞLARI TEMELLERİ	5
3.1. Petri Ağları Notasyonları ve Tanımları	5
3.2. Petri Ağları İşaretleri ve Durum Uzayı	7
3.3. Petri Ağı Dinamikleri	8
3.4. Temel Petri Ağı Yapıları	12
3.5. Durum Denklemleri	12
3.6. Petri Ağlarının Analizi	14
3.6.1. Güvenlik (Safeness) Özelliği	15
3.6.2. Sınırlılık (Boundness) Özelliği	15
3.6.3. Canlılık (Liveness) Özelliği	15
3.6.4. Sakınım (Conservation) Özelliği	17
3.7. Petri Ağlarının Zamanlaması	17
4. ARAÇ YÖRÜNGELERİ İÇİN PETRİ AĞLARI UYGULAMALARI	19
4.1. Otoyol Giriş Denetimi'nde Petri Ağı Uygulaması	19
4.1.1. Otoyol Giriş Denetimi	19
4.1.2. Giriş Denetiminin Üstünlükleri ve Sakıncaları	21
4.1.3. Otoyol Giriş Denetimi'nde Petri Ağı Uygulaması	23
4.1.3.1. Amaçlananlar	23
4.1.3.2. Kurallar	24
4.1.3.3. Amaçlanan Petri Ağlarının Tasarımı ve Analizi	27
4.2. Akıllı Araba ve Akıllı Yollar için Petri Ağı Uygulaması	51
4.2.1. Giriş	51

4.2.2 IVHS Görevleri	51
4.2.3 Tam Otomatikleştirilmiş Sistem	53
4.2.3.1 Takım Kavramı	53
4.2.3.2 Bir IVHS Senaryosu	55
4.2.3.3 Kontrol Sistemi Yapısı	57
4.2.4 IVHS'in Planlama Katmanı İçin Petri Ağı Uygulaması	59
5. SONUÇ	77
KAYNAKLAR	78
EKLER	79
ÖZGEÇMİŞ	86



KISALTMALAR

DES	: Discrete Event Systems
DEDS	: Discrete Event Dynamic Systems
CVDS	: Continuous Variable Dynamic Systems
FIFO	: First In First Out
IVHS	: Intelligent Vehicle / Highway System



TABLO LİSTESİ

	<u>Sayfa No</u>
Tablo 4.1. Yeşil ışık süresinin ayarlama kuralları.....	25
Tablo 4.2. Uyarı ekranında gösterilecek olan otoyol seçimi kuralları.....	26
Tablo 4.3. Trafik ışığı için kurallara göre yerler	29
Tablo 4.4. Örnek başlangıç durumlarına göre I1'in ışık süreleri	30
Tablo 4.5. Sürücü Kararları ve IVHS Görevleri.....	52
Tablo 4.6. Takımlarla Kapasite.....	54
Tablo 4.7. B'nin Birleşme Denemeleri Tablosu.....	66
Tablo 4.8. A aracına göre B'nin Birleşme Denemeleri Tablosu.....	75



ŞEKİL LİSTESİ

	<u>Sayfa No</u>
Şekil 3.1 : Tanımlara ilişkin örnek Petri ağı yapısı.....	6
Şekil 3.2 : Şekil 3.1'deki Petri ağının x_1 ve x_2 işaretlemeli halleri.....	7
Şekil 3.3 : $x_0 = [2, 0, 0, 1]$ durumlu Petri ağı.....	10
Şekil 3.4 : $x_1 = [1, 1, 1, 1]$ durumlu Petri ağı.....	10
Şekil 3.5 : $x_2 = [1, 1, 0, 2]$ durumlu Petri ağı.....	11
Şekil 3.6 : $x_2' = [0, 1, 0, 0]$ durumlu Petri ağı.....	11
Şekil 3.7 : Temel petri ağı yapıları.....	12
Şekil 3.8 : Örnek 3.4 için Petri ağı yapısı.....	16
Şekil 3.9 : Petri ağının zamanlaması.....	18
Şekil 4.1 : Denetim için algılayıcıların yerleri	20
Şekil 4.2 : Petri Ağı uygulamasının senaryo şekli.....	23
Şekil 4.3 : Trafik ışığı için Petri Ağı	28
Şekil 4.4 : Trafik ışığı için Petri Ağındaki Zamanlı Geçişler.....	31
Şekil 4.5 : Trafik ışığı Petri Ağı için başlangıç durumu.....	32
Şekil 4.6 : Trafik ışığı Petri Ağı için ikinci aşama.....	33
Şekil 4.7 : Trafik ışığı Petri Ağı için üçüncü aşama.....	34
Şekil 4.8 : Trafik ışığı Petri Ağı için dördüncü aşama.....	35
Şekil 4.9 : Trafik ışığı Petri Ağı için son aşama.....	36
Şekil 4.10 : Uyarı Ekranı ile ilgili Petri ağı.....	43
Şekil 4.11 : Uyarı Ekranı ile ilgili Petri ağı başlangıç durumu.....	45
Şekil 4.12 : Uyarı Ekranı ile ilgili Petri ağı 2. aşama.....	46
Şekil 4.13 : Uyarı Ekranı ile ilgili Petri ağı son aşama.....	47
Şekil 4.14 : Takım Kavramı.....	54
Şekil 4.15 : Bir güzergahın bir dizi kısımdan meydana gelmesi.....	55
Şekil 4.16 : Bir aracın planı.....	56
Şekil 4.17 : Kontrol Sistem Yapısı.....	57
Şekil 4.18 : Üç manevra.....	59
Şekil 4.19 : Birleşme için Durum Yapıları.....	62
Şekil 4.20 : B aracı için petri ağı ve sistem matrisi A.....	64
Şekil 4.21 : B aracının 4.denemesi için petri ağının 1. aşaması.....	67
Şekil 4.22 : B aracının 4.denemesi için petri ağının 2. aşaması.....	68
Şekil 4.23 : B aracının 4.denemesi için petri ağının 3. aşaması.....	69
Şekil 4.24 : B aracının 4.denemesi için petri ağının 4. aşaması.....	70
Şekil 4.25 : B aracının 4.denemesi için petri ağının 5. aşaması.....	71
Şekil 4.26 : B aracının 4.denemesi için petri ağının 6. aşaması.....	72
Şekil 4.27 : B aracının 4.denemesi için petri ağının son aşaması.....	73
Şekil 4.28 : A aracı için petri ağı.....	74
Şekil A.1 : PIPE Ekran Görüntüsü.....	79
Şekil B.1 : Simulaworks Ekran Görüntüsü.....	81
Şekil C.1 : Visual Objects Net++ Ekran Görüntüsü.....	83

SEMBOL LİSTESİ

e	: Olay
E	: Olay kümesi
T	: Sonlu geçiş kümesi
P	: Sonlu geçiş kümesi
A	: Oklar kümesi
w	: Okların ağırlık kümesi
$I(t_j)$	: t_j geçişine olan giriş
$O(t_j)$	: t_j geçişinden olan çıkış
$I(p_i)$	: p_i yerine olan giriş
$O(p_i)$	: p_i yerine olan çıkış
x	: Petri ağında işaretleme ile ilgili satır vektörü
L	: $T \rightarrow E$ geçiş etiketleme fonksiyonu
A_{ji}	: Petri ağlarının $(m \times n)$ boyutlu sistem matrisi
v	: Hız
Φ	: Araç /şerit / dakika cinsinden akış
d	: Öndeki araç ile olan (takımlar içi) mesafe
s	: Araç uzunluğu
D	: Takımlar arası mesafe
R	: Aracın yörüngesi (güzergahı)
N	: Hedef takım büyüklüğü

ÖZET

Bu tez çalışmasının amacı, Araç Yörüngeleri ile ilgisi bulunan 'Otoyol Giriş Denetimi' ve 'Akıllı Araç / Otoyol Sistemi (IVHS)' konularını araştırmak ve bu iki konu hakkında Ayrık Olay Sistem (DES) yöntemlerinden olan Petri ağları ile birer uygulama geliştirmektir. Otoyol Giriş Denetimi, otoyola konulan algılayıcılardan alınan bilgilerle araç akışının tespit edildiği ve araçların belli bir düzende otoyola alındığı bir denetleme sistemidir. Akıllı Araç / Otoyol Sistemi ise otoyola ve araçlara yerleştirilen kontrol, iletişim ve hesaplama teknolojilerinin bir birleşimi sayesinde sürücü kararlarında otoyol kapasitesini ve güvenliğini yeni yollar inşa etmeden arttıracak şekilde yardımcı olan bir sistemdir. Tezde ilk önce olay kavramından bahsedilmiş, Ayrık Olay Sistemleri hakkında bilgi verilmiş, Ayrık Olay Sistem yöntemlerinden biri olan Petri Ağı yapıları incelenmiş ve bu yapılarla ilgili örnekler verilmiştir. Daha sonra Petri Ağı Analizleri üzerinden durulmuştur. Sonrasında 'Otoyol Giriş Denetimi' hakkında bilgi verilmiş, üstünlükleri ve sakıncalarından bahsedilmiş ve bu konu ile ilgili bir petri ağı uygulaması gerçekleştirilmiştir. Bu uygulamada 'Otoyol Giriş Denetimi' haricinde aynı zamanda araç sürücülerine açık olan otoyolu seçmeleri için yol durumu hakkında bilgi verilmesine de çalışılmıştır. Daha sonra 'Akıllı Araç / Otoyol Sistemi' konusu incelemeye alınmış, sistemdeki katmanlar, sistemin kontrol yapısı, araçların sistemdeki manevraları ve birbirleriyle mesaj alışverişinde bulunmaları hakkında bilgi verilmiştir. En son olarak da 'Akıllı Araç / Otoyol Sistemi' ile ilgili olarak Simulaworks, PIPE, Visual Objects Net++ paket programları yardımıyla dört adet petri ağı uygulaması gerçekleştirilmiştir, programlardaki "jeton-oyun animasyonları" ile ağların işleyişleri gözlemlenmiş ve ağlardan ikisinin Python dilinde yazılmış bir programla analizi gerçekleştirilmiştir.

DISCRETE EVENT SYSTEM APPLICATIONS IN VEHICLE ROUTES

SUMMARY

The objective of this study is to research and make Discrete Event System (DES)-Petri net applications on two areas : "Highway Entrance Control" and "Intelligent Vehicle / Highway System (IVHS)" which are concerned with the "Vehicle Route" subject. Highway Entrance Control is a control system that collects information about vehicle flow from the sensors located on the highway and controls the vehicle flow in a proper regulation. "Intelligent Vehicle / Highway System (IVHS)" is such a system that a proper combination of control, communication and computing technologies placed on the highway and on the vehicle can assist drivers in ways that will increase highway capacity and safety without building more roads. Firstly we give information about Petri nets which is a language of Discrete Event Systems and present Petri net architectures and examples about them. After that, we focused on Analysis of Petri Net. Then, we give information about the subject "Highway Entrance Control" and state the advantages and disadvantages about it and make a Petri net application. In this application we also give information to the drivers about current traffic to choose the open highway. Then, we research "Intelligent Vehicle / Highway System (IVHS)" and give information about the concepts and control architecture of the system, vehicle maneuvers and message exchanges. Finally, by using Simulaworks, PIPE, Visual Objects Net++ petri net tools, four Petri net applications on "Highway Entrance Control" and IVHS was made and running of these applications were observed by "token-game animation"s of the tools and the analysis of two of these petri net applications were made by a programme written in Python programming language.

1. GİRİŞ

1.1. Giriş ve Çalışmanın Amacı

Araç Yörüngeleri kavramı bu tez çalışmasında , 'bir aracın izleyeceği yol, güzergah anlamında' ele alınmıştır. Ve bu amaçla çalışmada, otoyollardaki araçlarla ilgili uygulamalara değinilmiştir. Bunlardan birincisi 'Otoyol Giriş Denetimi' dir. Bu denetimde, otoyola konulan algılayıcılardan alınan bilgilerle araçların akışı tespit edilir ve araçlar belli bir düzende otoyola alınır. Diğer uygulama 'akıllı araç / otoyol sistemi' IVHS (Intelligent Vehicle / Highway Sistem) ile ilgilidir. IVHS, özel dizayn edilmiş araçların özel yapılmış otoyollarda, tek bir merkezden komuta edilerek sürücüsüz olarak gitmesini ve bu sayede trafik sıkışıklığına ekstra yollar yapılmaksınız bir çözüm sunma hedefinde bir sistemdir.

Her iki uygulamada da trafikle alakalı olduğu için bir Ayrık Olay Sistem örneğidir. Ayrık Olay Sistemi (DES), olayların düzensiz aralıklarla meydana gelmesi ile zaman içinde gelişen ancak dinamik yapısındaki gelişimleri olaylara bağlı olan sistemlerdir. Otoyol Giriş Denetimi'nde otoyola giren araçların düzensizliği ve rastgeleliği ile IVHS sistemindeki araçların trafikteki düzensizliği ve rastgeleliği bu uygulamalarda Ayrık Olay Sistemi kriterleri kullanımının faydalı olacağını göstermektedir. Bu amaçla, bu çalışmada, bir DES modeli olan Petri Ağları incelenmiş ve 'Otoyol Giriş Denetimi' ve IVHS için birer petri ağı uygulaması gerçekleştirilmiştir.

Sonuç olarak; bu çalışmanın amacı, Araç yörüngeleri ile ilgili olan 'Otoyol Giriş Denetimi' ve IVHS konularını incelemek ve bir DES modeli olan Petri ağları ile bu konularda uygulama gerçekleştirmektir. 2. Bölümde 'Ayrık Olay Sistemleri', 3.Bölümde 'Petri Ağları' incelenmiş, 4. Bölümde 'Otoyol Giriş Denetimi' ve IVHS incelenmiştir. 4. Bölüm'de Simulaworks ve Visual Objects Net++ paket petri ağı programları ile 'Trafik Işığı Petri Ağı' ve 'Uyarı Ekranı Petri Ağı' adlı iki petri ağı tasarlanmış ve bu ağların işleyişi, bu paket programlardaki "jeton-oyun animasyonu" ile gözlemlenmiştir. Ek olarak bu ağların Phyton programlama dilinde yazılmış bir analiz programı ile Sınırlılık, Canlılık, Sakınım özellikleri analiz edilmiş, ağlarının Ölü

Geçiş, Ulaşılabilirlik kontrolü de gerçekleştirilmiştir. 4.Bölüm’de akıllı araçların akıllı yollardaki haberleşmelerine örnek teşkil eden birbirleriyle içiçe iki petri ağı daha PIPE paket programı ile tasarlanmış ve içerisindeki “jeton-oyun animasyonu” ile bu iki ağın işleyişi incelenmiştir.



2. AYRIK OLAY SİSTEMLERİNE GİRİŞ

2.1. Olay Kavramı

Bir Ayrık Olay Sistemi, olayla yönlendirilen bir ayrık durum sistemidir. Bu sistemin durum evrimi, tamamen, zamanla meydana gelen asenkron ayrık olaylara bağlıdır.

Bir "olay"ın, aniden meydana geldiği ve bir ayrık durum değerinden diğer bir değere geçişe sebep olduğu düşünülür.

Örnekler :

- Butona basılır. (Bu, kontrol edilebilir bir olaya örnektir.)
- Bilgisayar birden çöker. (Bu, kontrol dışı bir olaya örnektir.)

2.1.1. Zaman Yürütümlü Sistem ile Olay Yürütümlü Sistemin Karşılaştırılması

Sürekli-durum sistemlerinde, durum genellikle zamanla değişir. Bu özel olarak ayrık zaman modelleri için de geçerlidir : "Saat" tipik bir örnek yolu sürendir. Her saat tik'inde durumun değiştiği varsayılır, yani sürekli durum değişkenleri zamanla sürekli değişirler. Bu yüzden bu sistemlere "zaman-yürütümlü" sistemler denir. Burada, zaman değişkeni (t sürekli zamanda ve k ayrık zamanda), tüm giriş, durum ve çıkış fonksiyonlarının argumanı olarak doğal, bağımsız bir değişkendir.

Ayrık-durum sistemlerinde durum sadece belli zaman noktalarında ani geçişlerle değişir. Bu gibi geçişlerin her birisine bir olay karşılık düşürdüğümüzü varsayalım. Ve zamanı ölçtüğümüz bir saatin var olduğunu düşünelim. İki adet olasılık vardır.

- 1- Her saat tik'inde bir olay e olay kümesi E'den seçilir. Eğer hiç bir olay meydana gelmez ise, bunu E'nin hiçbir durum değişikliğine yol açmayan "boş olay" elemanı olarak düşünebiliriz.
- 2- Çeşitli ani zaman değerlerinde (saat tiklerine denk düşmeyen ve tam olarak bilinmeyen), bazı e olayları meydana gelir.

Yukarıdaki 1 ve 2 arasında temel farklılıklar vardır. Bunlar :

- 1'de durum geçişleri saatle senkronizedir. Bir saat tik'i vardır, bir olay (ya da hiçbir olay) seçilir, durum değişir, ve işlem tekrarlar. Saat, tüm olası durum geçişlerinin tek sorumlusudur.
- 2'de her $e \in E$ olayı , e 'nin meydana geldiği andaki zaman değerleriyle farklı bir işlem tanımlar. Durum geçişleri, bu asenkron ve aynı zamana rastlayan olay işlemlerinin sonucudur.

Sonuç olarak, zaman - yürütümlü sistemlerde bütün durum geçişleri sistem saati ile senkronize edilir (Bir zaman yürütümlü sistem, sürekli de olabilir ayrık da). Zaman Yürütümlü sistemlere Sürekli Değişken Dinamik Sistemler - Continuous Variable Dynamic Systems (CVDS) denir.

Olay Yürütümlü sistemlerde durum geçişleri, asenkron ve rastlantısal olabilen olaylar tarafından gerçekleşir. Olay Yürütümlü sistemlere Ayrık Olay Dinamik Sistemler - Discrete Event Dynamic Systems (DEDS) ya da Ayrık Olay Sistemleri - Discrete Event Systems (DES) denir. [1]

2.2. Ayrık Olay Sistemleri

Ayrık olay sistemlerinde iki özellik bulunur :

1. Durum uzayı bir ayrık kümedir.
2. Durum geçiş mekanizması olay - yürütümlüdür.

Ayrık olay sistemleri, kuyruk sistemlerinde, bilgisayar sistemlerinde, iletişim sistemlerinde, üretim sistemlerinde, trafik sistemlerinde vs. kullanılır.

3. PETRİ AĞLARI TEMELLERİ

3.1. Petri Ağları Notasyonları ve Tanımları

Petri Ağı, Ayrık Olay Sistemleri için geliştirilmiş, olayları belli kurallar dahilinde kullanan bir araçtır. Petri ağları iki adımda açıklanabilir. İlk önce Petri ağı yapısı tanımlanır. Daha sonra bu yapıya başlangıç durumu, işaretli durumlar, geçiş fonksiyonları eklenerek bütün bir Petri Ağı modeli, ilgili dinamikleri ve ürettiği diller ve işaretlerle beraber verilir. [1]

Petri ağı yapısı, yerler ve geçişler olmak üzere iki tip düğüme ve bunları birbirlerine bağlayan oklara sahiptir. Oklar, aynı tip düğümleri birbirlerine bağlamazlar ya yer düğümünü geçiş düğümüne ya da geçiş düğümünü yer düğümüne bağlarlar.

Petri ağındaki geçişler ayrık olay sistemlerindeki olayları temsil ederler. Bir geçişin olması için incelenen sisteme ilişkin bazı şartların sağlanması gerekir. Bu şartlara ilişkin bilgiler *yerlerde* bulunur. Bu gibi bazı *yerler*, bir geçişe "giriş" olarak düşünülür ve bu geçişin sağlanabilmesi için gereken şartlar ile birlikte ilişkilendirilir. Diğer *yerler* bir geçişin "çıkışı" olarak düşünülür. Bunlar da geçiş sonucu oluşan şartlarla ilişkilendirilir. [1]

Bir Petri ağı 4 öğeden oluşmuştur. (P, T, A, w)

- P, sonlu yerler (mekan) kümesi.
- T, sonlu geçişler kümesi
- A, oklar kümesi ve $(P,T) \cup (T,P)$ kümesinin altkümesidir.
- w, bir ağırlık fonksiyonudur. $w : A \rightarrow \{ 1,2,3.. \}$
- Petri ağlarında olaylar, geçişler olarak düşünülür

Doğal olarak ilgilendiğimiz Petri ağının (P, T, A, w) yalıtılmış yer ve geçişleri olmadığını varsayıyoruz.

Oklar (p_i, t_j) ya da (t_j, p_i) olarak gösterilir. Okları ilişkilendireceğimiz ağırlıklar da tam sayı olmalıdır. Bu şekilde sınırlı sayıda yerler, geçişler ve oklar bir ayrık olay sistemini modellemek için yeterlidir. [1]

Petri ağı yapısında bir yerden t_j geçişine yapılan girişi $I(t_j)$ olarak, t_j geçişinden bir yere çıkışı da $O(t_j)$ olarak gösterebiliriz. Yani ;

$$I(t_j) = \{p_i \in P : (p_i, t_j) \in A\} \quad (3.1)$$

$$O(t_j) = \{p_i \in P : (t_j, p_i) \in A\} \quad (3.2)$$

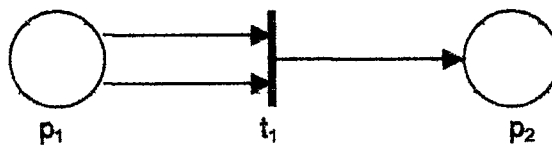
Aynı şekilde bir geçişten p_i yerine yapılan girişleri $I(p_i)$, çıkışları da $O(p_i)$ olarak gösterebiliriz. Petri ağı yapısını çizerken iki tip düğümü (geçiş ve yer) ayırt etmek için yerler daire ile geçişler de dik çizgi ile gösterilir. Yerleri ve geçişleri birbirine bağlayan oklar kümesi de A ile tanımlanır. P_i yerinden t_j geçişine çizilen bir ok $P_i \in I(t_j)$ olarak gösterilir. Eğer p_i yerinden t_j geçişine çizilen k adet ok varsa ya da k ağırlığında tek ok varsa bu da $w(p_i, t_j) = k$ olarak gösterilir. Aynı şekilde t_j geçişinden p_i yerine k ok bağlanırsa $P_i \in O(t_j)$ ve $w(t_j, p_i) = k$ ile ifade edilir. Petri ağı yapısında birden fazla oka bağlı olan düğümleri tek ok çizip üzerine ağırlığını yazarak göstermek uygun olur. Eğer ağırlığı belirtilmemişse 1 olduğu kabul edilir. Buradan şunu yazabiliriz[1] :

$$p_i \notin I(t_j) \text{ ise } w(p_i, t_j) = 0 \quad (3.3)$$

$$p_i \notin O(t_j) \text{ ise } w(t_j, p_i) = 0 \quad (3.4)$$

Örnek 3.1 : Bu tanımlardan aşağıdaki ifadelerle verilen petri ağı yapısını şu şekilde çizebiliriz[1] :

$$P = \{p_1, p_2\} \quad T = \{t_1\} \quad w(p_1, t_1) = 2 \quad w(t_1, p_2) = 1 \quad A = \{(p_1, t_1), (t_1, p_2)\}$$



Şekil 3.1 Tanımlara ilişkin örnek Petri ağı yapısı

3.2. Petri Ağı İşaretleri ve Durum Uzayı

Petri ağı yapısındaki geçişlerin ayrık olay sistemlerindeki olayları temsil etmesi için Petri ağındaki yerleri bu olayların meydana gelmesindeki şartlar olarak tanımlayabiliriz. Bu çerçevede şartların sağlanıp sağlanmadığını kontrol için bir mekanizmaya ihtiyaç vardır. Bu mekanizma yerlere jetonlar (token) atanarak sağlanabilir. Yerlere konulan jetonlar o yerin gerektirdiği şartlar olarak tanımlanır. Petri ağı yapısında yerlere atanan jetonlar işaretleme (marking) olarak tanımlanır. Bu özellikleri de göz önüne almak için Petri ağına ek büyüklüklere gerek olduğundan tanım aşağıdaki gibidir.

Tanım: Bir petri ağı yapısında (P, T, A, w) işaretli Petri ağını N ile gösterirsek

$$N = (P, T, A, w, E, L, X_0, X_m) \text{ olur.}$$

Burada ;

E : Geçişleri etiketlemek için olay kümesi

L : $T \rightarrow E$ geçiş etiketleme fonksiyonu

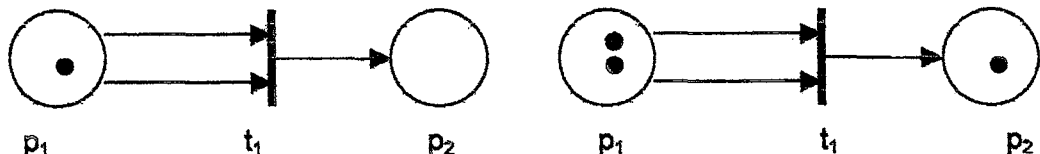
$X_0 \in N^n$ ağıın ilk durum kümesi (her yerdeki jetonların ilk sayısı)

$$X: P \rightarrow N = \{ 0, 1, 2, \dots \} \quad (3.5)$$

$$X = [x(p_1), x(p_2), x(p_3), \dots, x(p_n)] \quad (3.6)$$

$X_m \subseteq N^n$ ağıın işaretlenmiş durum kümesidir.

Petri ağındaki yer sayısı n olmak üzere işaretleme x satır vektörü olarak tanımlanır. Bu vektörün i. Elemanı p_i yerindeki jeton sayısını gösterir. Petri ağı yapısında bir jeton ilgili yerin üzerinde siyah bir nokta olarak gösterilir.



$$x_1 = [1, 0]$$

$$x_2 = [2, 1]$$

Şekil 3.2 Şekil 3.1'deki Petri ağıının x_1 ve x_2 işaretleme halleri

Jetonların Petri ağı içindeki dağılımı petri ağının o anki durumunu tanımlar. Yerlere atanan jeton sayısı sıradan pozitif tamsayılardır. N yere sahip bir petri ağının durum uzayı pozitif tam sayı elemanlı n boyutlu vektörler ile ifade edilebilir. $X = N^n$

Buraya kadar olan tanımlamalar Petri ağlarının durum geçiş mekanizmasını tanımlamaya yeterli değildir. Ayrık olay sistemini Petri ağı ile modelleyebilmek için bir notasyona daha ihtiyaç vardır. Petri ağı yapısındaki durum geçiş mekanizması için 'olanaklı geçişler' kavramı da kullanılmalıdır. [1]

Tanım : Olanaklı geçiş

Petri ağındaki bir geçişin $t_i \in T$, mümkün olabilmesi için ;

$$\text{Her } p_i \in I(t_i) \text{ için } x(p_i) \geq w(p_i, t_i) \quad (3.7)$$

şartı sağlanmalıdır. Yani t_i geçişinin yapılabilmesi için p_i 'deki jeton sayısı en az p_i 'yi t_i 'ye bağlayan ok sayısı kadar olmalıdır (t_i 'ye girişi olan tüm p_i ' ler için). Önceden belirtildiği gibi bir geçişin olabilmesi için gerekli şartlar yerler ile ilişkilendirilmiştir. Bütün şartlar sağlanınca bu geçiş mümkün olur. Jetonlar bu şartların sağlanması sürecinde kullanılan karar verme mekanizmalarıdır. Bir petri ağındaki olanaklı geçişler o otomasyonun durumundaki aktif olay kümesine eşittir. [1]

3.3. Petri Ağı Dinamikleri

Petri ağında durum geçiş mekanizması jetonların hareket ettirilmesiyle sağlanır. Bu şekilde Petri ağının durumu değişir. Bir geçiş mümkün olduğunda buna ateşleyebilir ya da oluşabilir denir (Petri ağları literatüründe ateşleyebilir (firing) terimi kullanılır). Petri ağının durum geçiş fonksiyonu oluşabilecek olanaklı geçişlerin durumuna göre tanımlanır[1].

Tanım : Durum geiş fonksiyonu

(P, T, A, w) Petri ağıının $t_j \in T$ geişı için durum geiş fonksiyonu $f : N^n \times T$, aşığıdaki koşul altında

$$\text{Her } p_i \in I(t_j) \text{ için } x(p_i) \geq w(p_i, t_j) \quad (3.7)$$

tanımlanabilir. Eğer $f(x, t_j)$ tanımlanırsa $x' = f(x, t_j)$ Őu Őekilde olur : $i = 1, 2, \dots, n$ olmak üzere

$$x'(p_i) = x(p_i) - w(p_i, t_j) + w(t_j, p_i) \quad (3.8)$$

(3.8)'den anlaşılaçağı gibi durum geiş fonksiyonu sadece olanaklı geişler için tanımlanabilir. Bu da otomasyondaki műmkűn olay ifadesine eşıttir. (3.8)'den gűrűleceğı gibi bir sonraki durumla ilgili geişin giriő ve ıkıő yerlerine ve bu yerlerden geiőe baėlanan okların ağırlıėına baėlıdır. Eğer p_i, t_j geiőine giriő yeri ise p_i 'yi t_j 'ye baėlayan okun ağırlıėı kadar p_i jeton kaybeder. Eğer p_i, t_j 'nin ıkıő yeri ise t_j 'yi p_i 'ye baėlayan okun ağırlıėı kadar p_i jeton kazanır. Petri ağıında jeton sayısı geiőler sonucu korunmayabilir [1].

(3.8)'den gűrűleceğı gibi aşığıdaki eőitsizlikler műmkűndűr :

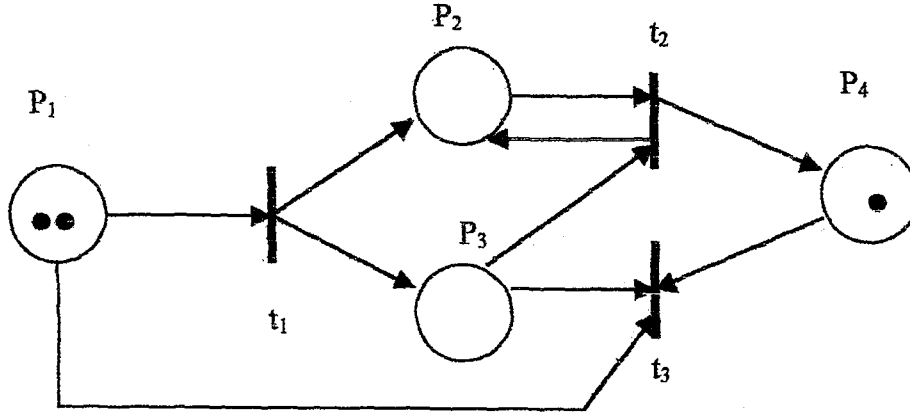
$$\sum_{p_i \in P} w(t_j, p_i) > \sum_{p_i \in P} w(p_i, t_j) \quad (3.9)$$

$$\sum_{p_i \in P} w(t_j, p_i) < \sum_{p_i \in P} w(p_i, t_j) \quad (3.10)$$

Bu eőitsizlikler bir yerden bir yere geiőe izilen okların sayısının, aynı geiőten aynı yere izilen okların sayısından bűyűk ya da kűűk olabileceėini sűylemektedir.

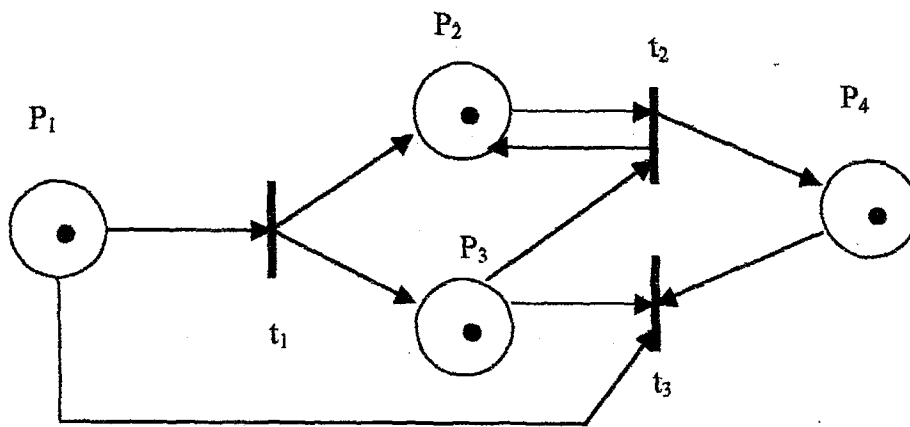
$x' = f(x, t_j)$, x' 'den daha ok ya da daha az jetona sahip olabilir. Genelde birkaç geiő sonunda $x = [0, 0, 0, \dots, 0]$ durumu da műmkűndűr ya da ok fazla geiő sonunda belli yerlerdeki jeton sayısı ok fazla artabilir.

Ŗrnek 3.2 : Bir petri ağıının durumunun geiőler sonucu deėiőmesini bir Ŗrnekle gűsterelim[1]. İlk durumu $x_0 = [2, 0, 0, 1]$ olan bir petri ağıını inceleyelim. Őekil 3.3'den gűrűldűėű gibi sadece t_1 geiőı műmkűndűr.



Şekil 3.3 $x_0 = [2, 0, 0, 1]$ durumlu Petri ağı

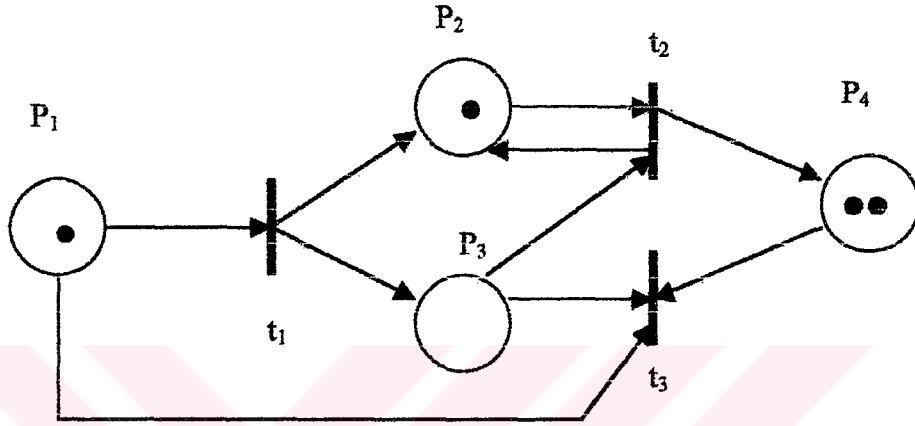
Çünkü geçiş için sadece bir jetonun geçmesine gerek vardır ve $x_0(p_1)=2$ 'dir. Yani $x(p_1) \geq w(p_1, t_1)$ ile (3.7)'deki koşul sağlanmaktadır. Petri ağında görüldüğü gibi t_1 geçişi yapıldığında p_1 'den bir jeton eksilir ve p_2 , p_3 yerlerine birer jeton yerleştirilir. (3.8) denklemi uygulandığında da $x_1 = [1, 1, 1, 1]$ durumu (Şekil 3.4) elde edilir.



Şekil 3.4 $x_1 = [1, 1, 1, 1]$ durumlu Petri ağı

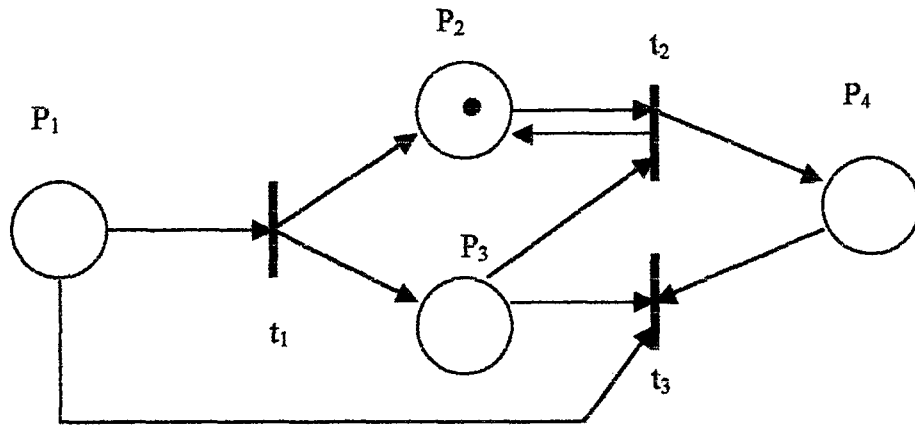
Bu durumda üç geiş de (t_1, t_2, t_3) olanaklıdır. Bir sonraki adımda t_2 geişi yapılırsa p_2 ve p_3 'den birer jeton taşınır. Çıkış yerleri p_2 ve p_4 'dür.

Bu şekilde bir jeton hemen p_2 'ye geri döner çünkü $p_2 \in I(t_2) \cap O(t_2)$ 'dir. Bir jeton da p_4 'e eklenir. Yeni durum $x_2 = [1, 1, 0, 2]$ (Şekil 3.5) olur.



Şekil 3.5 $x_2 = [1, 1, 0, 2]$ durumlu Petri ağı

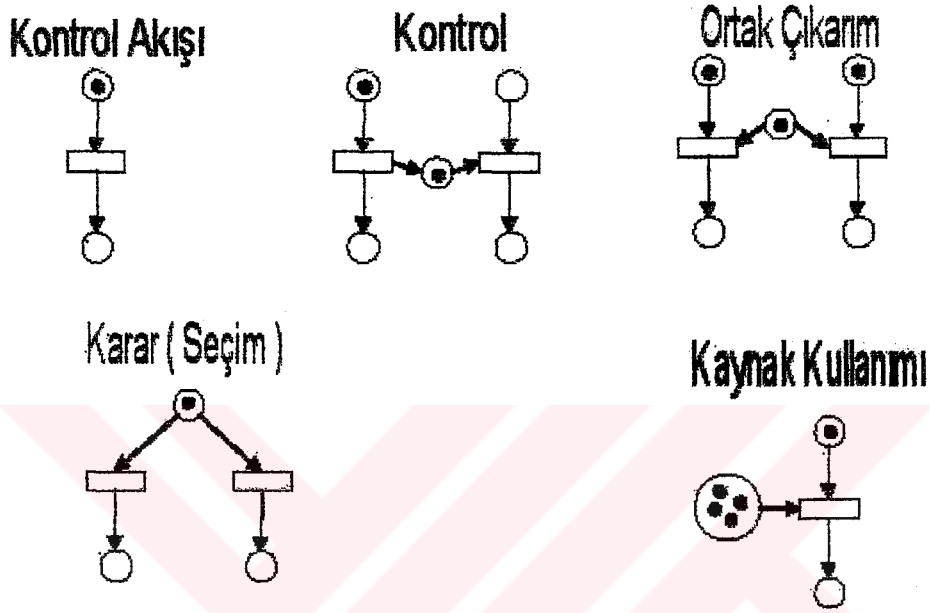
Bu durumda t_2 ve t_3 geişleri mümkün değildir. Sadece t_1 geişi olanaklıdır. Şekil 3.4'de üç geişin mümkün olduğu durumda t_2 geişi yerine t_3 geişinin yaparsak p_1, p_3, p_4 yerlerinden birere jeton taşınır. Görüldüğü gibi t_3 geişinden sonra çıkış yeri yoktur. Yeni durum $x_2' = [0, 1, 0, 0]$ (Şekil 3.6) şeklinde olur.



Şekil 3.6 $x_2' = [0, 1, 0, 0]$ durumlu Petri ağı

Bu durumda geiş artık olanaklı deėildir ve bařka durum deėiřimi olmaz. Yani $[0, 1, 0, 0]$ durumu Petri aėının ıkamazdır (deadlock). rnekte jeton sayısı da korunmamaktadır. X_0 durumu 3; x_1 ve x_2 durumları 4 ve x_2^1 durumu da 1 jetona sahiptir.

3.4 Temel Petri Aėı Yapıları



řekil 3.7 Temel petri aėı yapıları [2]

3.5. Durum Denklemleri

Durum deėiřimleri Petri aėları ile her bir duruma bir yeni yapı karřı dřrlerek nasıl ifade edilir bir nceki blmde incelendi. Bu gsterim grldė gibi ok pratik deėildir. Dinamik yapının incelenmesinde daha uygun bir yaklařım durum deėiřimini matematiksel olarak ifade etmektir. Bir yerin durum deėerinin bir geiř olduėunda nasıl deėiřtiėini 3.8 denkleminde elde edebiliriz. Bir t_j geiři ncesindeki durumu ;

$$X = [x(p_1), x(p_2), \dots, x(p_n)] \quad (3.11)$$

ile gsterirsek bir sonraki durum

$$X' = [x'(p_1), x'(p_2), \dots, x'(p_n)] \quad (3.12)$$

olur.

j. geçişin olduğu sıradaki durumu ifade etmek için m boyutlu bir ateşleme (firing) vektörü tanımlayabiliriz.

$$U=[0,...,0,1,...,0] \quad j \in \{1,2,...,m\} \quad (3.13)$$

Petri ağlarının (m x n) boyutlu sistem matrisi A ;

$$A_{ji} = w(t_j, p_i) - w(p_i, t_j) \quad (3.14)$$

olarak tanımlanır.[1]

Bu ifade 3.8 denklemindeki $x(p_i)$ 'nin ağırlık farkına benzerdir. Bu tanımlardan sonra vektör durum denklemini yazabiliriz.

$$x' = x + u.A \quad (3.15)$$

Bu ifade u girişi sonucunda durum geçiş işlemini tanımlar. Daha önce tanımlanmış olan $f(x, t_j)$ geçiş fonksiyonunun da

$$f(x, t_j) = x + u.A \quad (3.16)$$

olduğu görülür. Fonksiyonda t_j ifadesi sıfırdan farklı olan u'daki j.değeri gösterir. Durum denklemleri geçiş işlemi sonucunda değişen Petri ağının durumunu tanımlamaya yarar.

Örnek 3.3 : Şekil 3.3'de verilen, ilk durumu $x_0 = [2, 0, 0, 1]$ olan Petri ağını inceleyelim. 3.14'deki ifade ile sistem matrisi

$$A = \begin{bmatrix} -1 & 1 & 1 & 0 \\ 0 & 0 & -1 & 1 \\ -1 & 0 & -1 & -1 \end{bmatrix} \quad \text{olur.}$$

3.16 denklemleri kullanılarak t_1 geçişi oluştuğu zaman $u = [1 \ 0 \ 0]$ olur ;

$$\begin{aligned}
x &= [2 \ 0 \ 0 \ 1] + [1 \ 0 \ 0] \begin{vmatrix} -1 & 1 & 1 & 0 \\ 0 & 0 & -1 & 1 \\ -1 & 0 & -1 & -1 \end{vmatrix} \\
&= [2 \ 0 \ 0 \ 1] + [-1 \ 1 \ 1 \ 0] \\
&= [1 \ 1 \ 1 \ 1]
\end{aligned}$$

elde edilir. Sonucun bir önceki örnekle aynı olduğu görülür. Aynı şekilde t_2 geçişi sonucunda elde edilen x_2 durumu ;

$$\begin{aligned}
x_2 &= [1 \ 1 \ 1 \ 1] + [0 \ 1 \ 0] \begin{vmatrix} -1 & 1 & 1 & 0 \\ 0 & 0 & -1 & 1 \\ -1 & 0 & -1 & -1 \end{vmatrix} \\
&= [1 \ 0 \ 1 \ 2]
\end{aligned}$$

olur.

Görüldüğü gibi Petri ağında izlenen yol, geçiş sırasına göre $\{t_1, t_2, \dots\}$ (t^k k. oluşan geçiş) durumlarının sıralanmasıdır. $\{x_0, x_1, \dots\}$

x_0 durumu verildiğinde sırayla tüm durumlar :

$$x_{k+1} = f(x_k, t_k) = x_k + u_k A \quad (3.17)$$

denklemleri ile üretilebilir. (u_k , k. geçişin ateşlendiğini belirtir) [1].

3.6. Petri Ağlarının Analizi

Petri ağlarıyla modellenen bir sistemin analizi de gereklidir. Modellenen sistemin davranışlarını inceleme açısından bu analizler çok önemlidir. Petri ağlarının analizleri için birçok teknik geliştirilmiştir fakat birçok problem de hala çözülememiştir.

3.6.1. Güvenlik (Safeness) Özelliği

Petri ağında bir yerdeki jeton sayısı 1'i kesin geçmiyorsa o yer güvenlidir. O yerdeki jeton sayısı ya 1 ya da 0'dır. Bu şekilde yerler mantıksal bir durumu ifade edebilir (Jeton varsa doğru yoksa yanlış şeklinde)[1].

Tanım : Petri ağındaki bir yer $p_i \in P$, $N = (P, T, I, O)$ eğer tüm yeni durumlar için $x' \in R(N, x)$ $x'(p_i) \leq 1$ ise, ilk durumu x ile güvenlidir.

Bir petri ağındaki tüm yerler güvenli ise o Petri ağı güvenlidir [1].

3.6.2. Sınırlılık (Boundedness) Özelliği

Klasik sistem teorisinde bir durum değişkeninin çok fazla büyümesi o sistemdeki kararsızlığa işaret eder. Aynı şekilde Petri ağının bir yerinde biriken jetonlar sistemi kararsızlığa götürür. Sınırlılık özelliği bir yerdeki jeton sayısının kesinlikle pozitif bir tam sayıyı geçemeyeceğine işaret eder [1].

Tanım : N Petri ağında x_0 başlangıç durumundaki $p_i \in P$ yeri eğer ulaşılabilir tüm durumlar için $x \in R(N)$, $x(p_i) \leq k$ ise k-sınırlıdır denir [1].

Eğer Petri ağındaki tüm yerler sınırlı ise o ağa "sınırlıdır" denir. Eğer bir yer "1-sınırlı" ise o yer aynı zamanda güvenlidir. Petri ağı ile modellenen bir ayrık olay sisteminde ağın sınırlı olup olmadığı kontrol edilir ve bir sınırlılığa karar verilir. Eğer sınırlılık sağlanmıyorsa model değiştirilmelidir. Çünkü kararsızlığa neden olabilir [1]

3.6.3. Canlılık (Liveness) Özelliği

Genellikle petri ağındaki geçişlerin ateşlenip ateşlenemeyeceği ile ilgilidir. Eğer geçiş hiç ateşlenmeyecekse ilgili olay meydana gelmeyecek demektir. Bu aynı zamanda kilitlenme(deadlock) özelliği ile de ilgilidir. [1]

Tanım : X_0 başlangıç durumlu bir N petri ağında, x_0 durumundan ulaşılan yeni durumlarda her zaman canlanabilir bir geçiş bulunuyorsa o ağa "canlıdır" denir[1].

Bu özellik sistemin davranışıyla ilgili önemli bir durumdur.

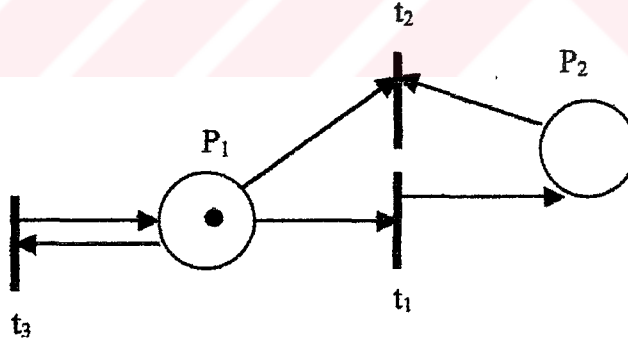
Canlılık ayrıca 4 seviyede sınırlandırılır [1].

x_0 başlangıç durumundaki petri ağındaki bir geçiş :

- Eğer x_0 durumundan hiç ateşlenmiyorsa ölü ya da L0-canlıdır.
- Eğer x_0 durumundan en az 1 kez ateşleniyorsa L1-canlıdır.
- Eğer en az k kere ateşlenebiliyorsa L2-canlıdır.
- Eğer sonsuz ateşlenebiliyorsa L3-canlıdır.
- Eğer x_0 durumundan ulaşılabilen tüm durumlar için L1-canlı ise buna L4 canlıdır.

denir.

Örnek 3.4 Şekil 3.8'deki Petri ağı yapısında t_2 geçişi ölüdür. Çünkü sadece p_1 ve p_2 'de aynı anda birer jeton bulunduğu anda ateşlenebilir. Fakat bu hiçbir zaman gerçekleşmez (Eğer t_1 ateşlenirse p_2 bir jeton alır ama p_1 jetonunu kaybeder). t_1 geçişi L1-canlıdır. Çünkü sadece bir kez ateşleyebilir. Gerçekte eğer t_1 ateşlenirse ondan sonra tüm geçişler ölü hale gelir. Son olarak, t_3 geçişi L3-canlıdır çünkü sonsuz sıklıkta ateşlenme olasılığına sahiptir. Fakat t_1 ateşlendiğinde ölü konuma geldiği için L4-canlıdır. [1]



Şekil 3.8 Örnek 3.4 için Petri ağı yapısı

3.6.4. Sakınım (Conservation) Özelliği

Bazen jetonlar, bir kaynak paylaşma sisteminde, müşterilerden ziyade kaynakları temsil ederler. Verilen bir ayrık olay sistemi petri net modelinde, sık sık, bu kaynakların kaybedilmediğini ya da kaynak kazanılmadığını gösteren kesin ağırlıklarla ilişkili sakınımın sağlanması gerekir.

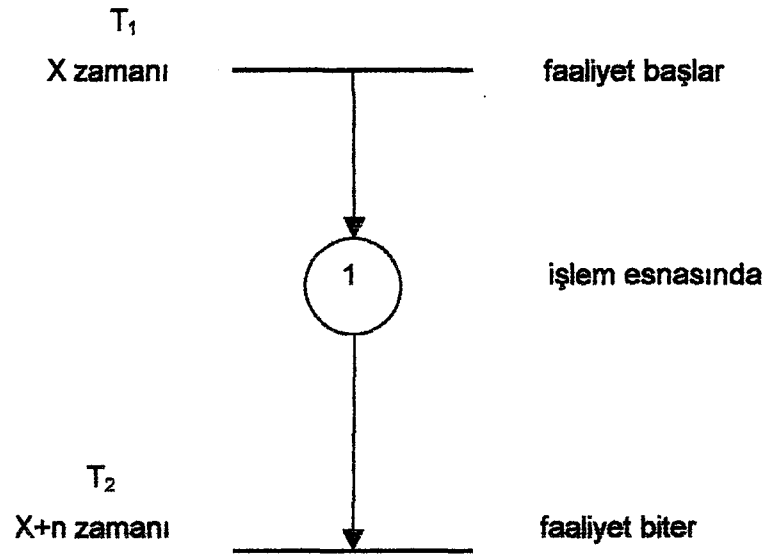
Tanım : N Petri ağında eğer her $x \in R(N)$ için

$$\sum_{i=1}^n \gamma_i x(p_i) = \text{sabit} \quad (4.1)$$

ise x_0 ilk durumu, " $\gamma = [\gamma_1, \gamma_2 \dots \gamma_n]$ "'e bağlı olarak sakınımlıdır" denir.[1]

3.7 Petri Ağlarının Zamanlaması

Petri ağlarının zamanlaması şu şekilde gerçekleştirilir : Tamamlanması için belli bir süre (n zaman birimi olsun) geçen bir faaliyetin petri ağı ile temsil edildiğini düşünelim. Faaliyet iki geçiş ve bir şart ile modellenebilir. Bu zaman faktörü, faaliyetin başlangıç zamanı (x zaman birimi) ve sonrasında faaliyet sonu için olan zaman ($x+n$) zaman birimi olarak düşünülebilir. Bu yüzden ilk geçiş (faaliyetin başlangıcı) x zamanında meydana gelecektir ve bu, ikinci geçişe (faaliyet bitişi) $x+n$ zamanında ateşlenebilmesi için bildirilecektir (Şekil 3.9) [3].



Şekil 3.9 Petri ağıının zamanlaması

4. ARAÇ YÖRÜNGELERİ İÇİN PETRİ AĞI UYGULAMALARI

4.1. Otoyol Giriş Denetimi'nde Petri Ağı Uygulaması

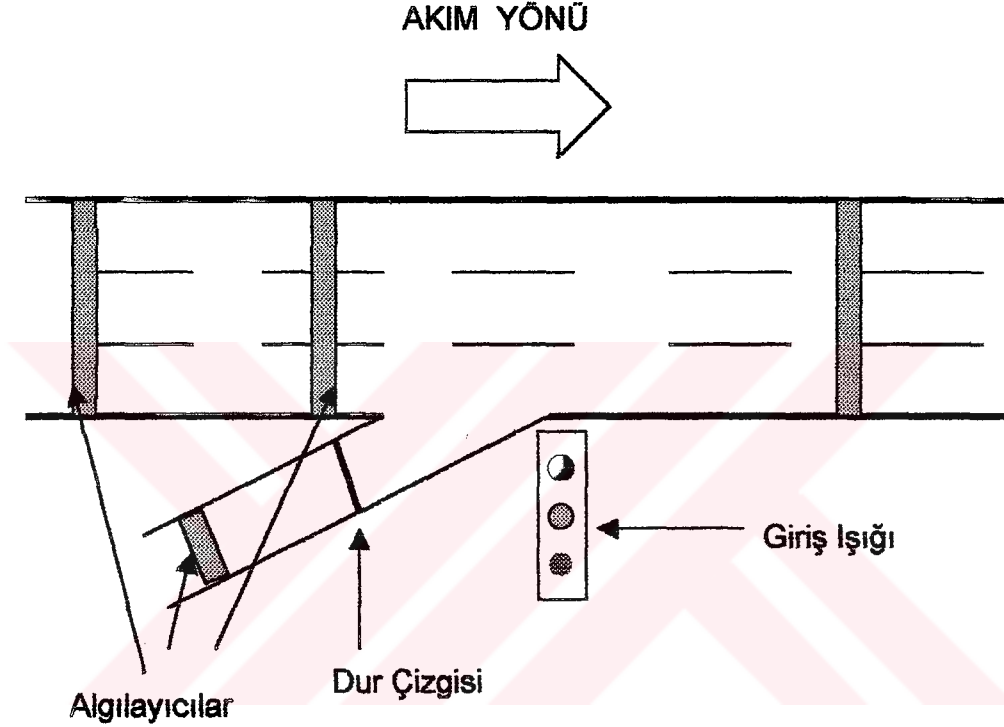
4.1.1. Otoyol Giriş Denetimi

Şehir içi caddelerde trafik akımı (trafik ışıkları vasıtasıyla) gruplandırılarak yönetilir. Şehir içi otoyollarda ise gruplanma istenmez. Çünkü oluşacak hız farkları altyapı kapasitesinin kullanımını azaltır. Altyapı kapasitesini en uygun şekilde kullanacağından otoyol akımında kararlı akım şartları hüküm sürmelidir.

Giriş denetimi yeni bir trafik yönetim kavramı değildir; kırk yıldır kullanılan etkin bir denetim aracıdır. Giriş denetiminin basit özeti, otoyol girişine yerleştirilecek trafik ışığı sayesinde girişteki araçların otoyola kontrollü katılmalarının sağlanmasıdır. Otoyol giriş denetimi, kapasite paylaşım yöntemlerinden birisidir. Uygulama, otoyol girişlerinde araçların otoyola erişimini denetleyerek akıcı olan otoyola giren araçları sınırlandırarak, giren trafik hacmini denetleyerek ve giren taşıtları aralıklara bölerek otoyolun kapasitesinin verimli kullanılmasını sağlar. Giriş denetimi uygulayarak , dur kalk gecikmelerinin azaltılması, planlı güzergah değişimlerinin özendirilmesi ve altyapının istenen hizmet düzeyine ulaştırılması amaçlanır. Uygulamanın başarısı, anayol akımının hızını düşürmeden, akıma katılabilecek en yüksek hacimden sonra otoyol girişi hacimlerini sınırlandırmasına, düzenlemesine ve taşıtların uygun zamanlamayla katılmalarına izin vermesine bağlıdır[4].

Giriş denetim uygulaması, " ilk gelen ilk hizmet alır" (FIFO : first in first out) prensibine dayanarak otoyolun artık kapasitesini girişteki araçlara ayırmaktır. Girişe gelen araçlar belirli bir süre veya anayol akımı belli bir hacim değerine düşünceye kadar bekletilirler. Artık kapasite anayoldaki araçların çıkışlardan anayolu terk etmeleri ile gerçekleşir.

Giriş denetimi için otoyol girişinde trafik ışığı, giriş yolu ile otoyolda algılayıcılar ve denetim donanımı gereklidir (Şekil 4.1). Otoyol algılayıcılarından gelen bilgi, denetim donanımı ile işlenir, otoyolun artık kapasitesi bulunur, trafik ışığı ve girişteki algılayıcı sayesinde artık kapasite varsa, giriş yolundan aracın anayola katılmasına izin verilir.



Şekil 4.1 Denetim için algılayıcıların yerleri

Basit bir giriş denetimi için üç adet algılayıcı gereklidir : Anayol sağ şeritte aralık algılayıcısı, giriş yolunda talep algılayıcısı ve otoyol ana algılayıcısı. Aralık algılayıcısı, anayol en sağ şeridinde taşıtlar arasındaki aralıkları ölçen ve otoyol trafik ışığı için gerekli bilgiyi aktaran ana algılayıcıdır. Talep algılayıcısı, giriş yolunda dur çizgisi arkasında bekleyen araç sayısını belirleyen algılayıcıdır [4] .

Öncelikle hesaplanmış otoyol kapasitesi ile anayol algılayıcısı sayesinde ölçülen yukarı akım hacminin farkı (artı kapasite) bulunur. Artık kapasite aynı zamanda otoyola katılabilecek araç sayısıdır ve belirlenecek zaman aralıkları boyunca güncellenir. Aralık algılayıcısı sayesinde araçlar arası uygun boşluk bulunur. Bu boşluk için hesaplanan araç sayısına uygun yeşil ışık girişte yakılır ve denetim hacminin (otoyol girişinde denetimli erişimi sağlanan akımın) akıma katılması sağlanır.

4.1.2 Giriş Denetiminin Üstünlükleri ve Sakıncaları

Yapılan teorik araştırmalar ve bölgeler uygulamalar ortaya koymuştur ki giriş denetimi uygulaması

- a. Tekrar eden veya tekrar etmeyen otoyol tıkanıklarının büyüklüğünü azaltır
- b. Otoyoldaki bekleme sürelerini azaltarak toplam yolculuk süresini iyileştirir.
- c. Otoyolda hizmet verilen toplam araç sayısını arttırarak kapasite kullanım oranını yükseltir.

Giriş denetiminin otoyolda tıkanıkları önlerken veya azaltırken şehir içi yollara olumlu ve olumsuz etkileri olabilir. Otoyol girişi kuyruklanmaları sürücüler farklı güzergah seçmeye yönlendirebilir (ki bu şehir içi yollara talebi arttırır) veya otoyolda tekrar eden tıkanıklıkların azaltılması, önlenmesi daha fazla sürücünün otoyolu kullanmasını sağlayabilir (bu da şehir içi yollarda talebin azalmasına yol açar) .

Otoyol giriş denetimi, zirve saatlerde, artmayı sürdüren talebe karşı trafik akımını güzergah seçeneklerine kaydırarak rahatlama sağlar. Farklı güzergahlara yeterli araç kaydırılırsa, azalan talep otoyol tıkanıklıklarını azaltabilecek hatta ortadan kaldıracaktır. Bununla birlikte otoyol, kent ulaşım ağının bir parçası olduğu için seçenekler arası geçişi sağlayan yolların kapasiteleri bu güzergahlara kaydırılan trafiği taşıyabilecekse kaydırma yararlıdır. Aksi halde yapılan tek etki tıkanıklığın otoyoldan uzaklaştırılıp şehir içi caddelere kaydırılmasıdır.

Denetim yapılan girişte, tüm kullanıcılara eşit davranılmasına (yani otoyolda artık kapasite oldukça anayola "ilk gelen ilk hizmet alır" ilkesi uyarınca katılımlara izin verilmesine, denetim önceden ayarlı ise kullanıcıların eşit süre bekletilmesine) rağmen kullanıcıların zamana verdikleri değerlerinin farklı olması eşitsizlik oluşturmaktadır. Başka güzergah seçme şansı olmayan ve denetim yapılan zaman aralığında girişi kullanmak zorunda olan zaman değeri yüksek kullanıcılar için eşitsizlik söz konusudur.

Otoyola şehrin merkezi olmayan bölgelerinden giriş yapacak olanlar ya hiç beklemeyecek ya da merkezi bölgelerden giriş yapanlara göre daha az bekleyeceklerdir ve otoyolun tüm imkanlarından daha fazla yararlanacaklardır. Merkezi bölgelerden otoyola giriş yapanlar için bu da eşitsizlik olarak öne çıkmaktadır ve halkın uygulamayı kabulünü zorlaştırmaktadır.

Giriş denetiminde kırmızı ışık ihlallerini yüksek oranlara ulaşması uygulamanın yararını perdeleyecek diğer bir etkidir.

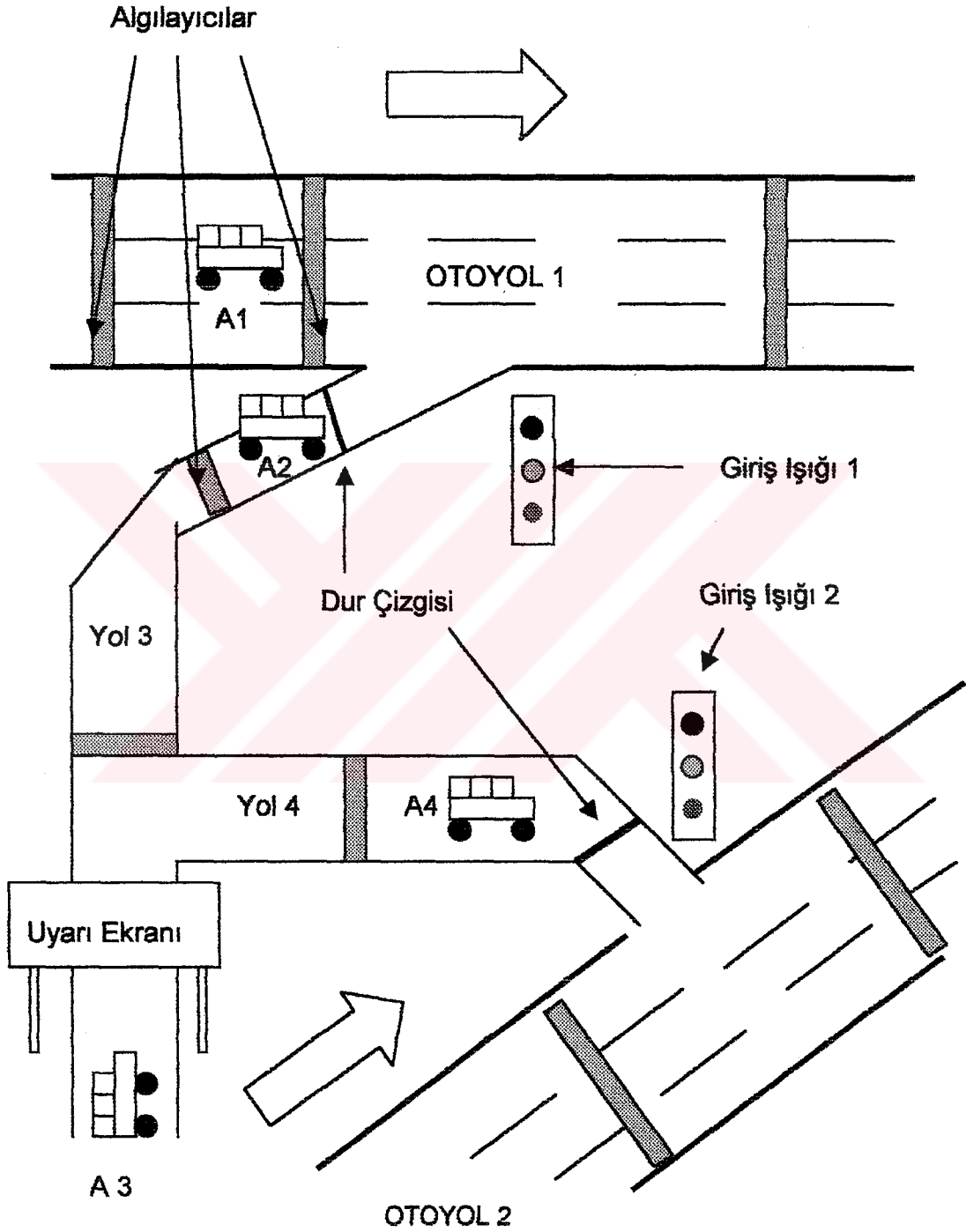
Denetim yapılan otoyol kesiminde doygun akım üç çeşit senaryo ile oluşabilir :

- 1- Günlük zirve saatlerde olduğu gibi, denetimli otoyol kesiminde otoyola giriş talebi, bu kesimi denetim süresince tıkayabilecek kadar yüksektir.
- 2- Otoyol kesimi kısadır, düşük giriş talebini yüksek yukarı akım engellemektedir ve tıkanıklık yüksek anayol talebinden oluşmaktadır.
- 3- Kontrol edilen kesim, ağır bir kaza sonucu oluşan şişe boynu bir kesitin hemen önündedir.

Bu üç durumun gerçekleşmesinin hemen ardından araçları geçici olarak otoyol girişinde bekletmek, otoyol sistemini tıkanıklıktan kurtarmayacak ancak otoyolda yolculuk süresini azaltacaktır. Aynı zamanda önlenemez olarak otoyol girişlerinde gecikme süreleri uzayacaktır. Bu uzaman anayol yolculuk süresinde meydana gelen azalmayı dengeleyecek hatta kendi yönünde bozacaktır. Bu yüzden tıkanıklıklar oluşmadan önlemlerin alınması en kısa zamanda ve en fazla araca hizmet vermek için gereklidir [4].

4.1.3. Otoyol Giriş Denetimi'nde Petri Ağı Uygulaması

4.1.3.1. Amaçlananlar



Şekil 4.2 Petri Ağı uygulamasının senaryo şekli

Şekil 4.2'de görüldüğü üzere Otoyol 1 ve Otoyol 2 ve yan yollar olan Yol 3 ve Yol 4'de akan araçlar için bir petri ağı uygulaması gerçekleştirilmiştir. Bu uygulamada amaçlananlar şunlardır :

- 1- Giriş Işığı 1'in yeşil yanma durumunu Otoyol 1 ve Yol 3'ün yoğunluğuna göre ayarlayarak A2 araçlarının otoyola girişini petri ağı ile denetlemek
- 2- Otoyol 1, Yol 3 ve Yol 4 'ün yoğunlukları karşılaştırarak o anki en açık yolun hangisi olduğunu Uyarı Ekranında A3 araçlarına bildirmeye yardımcı olacak bir petri ağı oluşturmak.

4.1.3.2. Kurallar

Yeşil ışığın yanma süresinin hesaplanması için algılayıcılar Otoyol 1'in ve Yol 3'ün akış yoğunluğunu kontrol merkezine bildirdiğini ve sayısal sınır değerlerine göre üç ayrı 'sözel değer'in ortaya çıktığını varsayıyoruz. Bu varsayım ile birlikte burada bir tür bulanık mantık [5] kullanmış oluyoruz. Buna göre ;

Akış Yoğunluğu → 'Az', 'Orta', ya da 'Çok' Yoğunluktadır.

Buna karşılık Giriş Işığında üç ayrı yeşil yanma süresi olduğunu varsayıyoruz. Bunlar :

Yeşil ışık süresi → 'Az' =(Örneğin 30sn),
'Orta'=(Örneğin 60sn.),
'Çok'=(Örn.90sn.)

Yukarıdaki varsayımları 1 ve 2. şıklarda tasarlanmasını amaçlanan petri ağında kullanabilmek için Tablo 4.1 ve 4.2'deki kurallar geliştirilmiştir. Buradaki bulanık mantık içeren kurallar tahmini olarak uzmana danışılmadan örnekte kullanılması için yazılmıştır. Otoyolların, akışın durumuna göre uzman bilgisi dahilinde değiştirilebilir (Örn. "Y1=A ve Y3 =O ise Ç " iken "Y1=A ve Y3=O ise O" kuralı yazılabilir).

Tablo 4.1 Yeşil ışık süresinin ayarlama kuralları

Y_1	VE	Y_3	ISE	I_1
A		A		O
A		O		Ç
A		Ç		Ç
O		A		A
O		O		O
O		Ç		Ç
Ç		A		A
Ç		O		O
Ç		Ç		O

Tablo 4.1’de ve 4.2’deki sembollerin anlamları şunlardır :

Y_1 = Otoyol1’in araç akış yoğunluğu,

Y_3 = Yol3’ün akış yoğunluğu,

Y_4 =Yol4’ün akış yoğunluğu

I_1 = Giriş ışığı 1’in yeşil ışık yanma durumu

A = Az süreli (ışık için), Az yoğun (yol için)

O = Orta süreli (ışık için), Orta yoğun (yol için)

Ç = Çok süreli (ışık için), Çok yoğun (yol için)

1 = Otoyol1 daha açık

2 = Otoyol2 daha açık

H = Her iki otoyol da yaklaşık olarak aynı yoğunlukta .

Örneğin Tablo 4.1’deki ilk satırı açıklarsak :

$Y_1 = A$ ve $Y_3 = A$ ise $I_1 = O$ ’dur. Yani, Otoyol 1’in akış yoğunluğu AZ iken Yol 3’ün akış yoğunluğu da AZ ise o halde Giriş Işığı 1’in yeşil ışık Yanma süresi ORTA(=60sn)dir.

Tablo 4.2'deki ilk satırı açıklarsak :

$Y_1 = A$ VE $Y_3 = \text{Ç}$ VE $Y_4 = A$ İSE OTOYOL SEÇİMİ = 2'dir. Yani Otoyol 1'in akış yoğunluğu AZ iken Yol 3'ün akış yoğunluğu ÇOK ve Yol 4'ün akış yoğunluğu AZ ise A3 araçları için seçilmesi daha uygun olan otoyol 2'dir.

Tablo 4.2 Uyarı ekranında gösterilecek olan otoyol seçimi kuralları

Y_1	VE	Y_3	VE	Y_4	İSE	OTOYOL SEÇİMİ
A		Ç		A	→	2
A		Ç		Ç	→	1
A		Ç		O	→	H
A		A		A	→	H
A		A		Ç	→	1
A		A		O	→	1
A		O		A	→	2
A		O		Ç	→	1
A		O		O	→	H
Ç		Ç		A	→	2
Ç		Ç		Ç	→	H
Ç		Ç		O	→	2
Ç		A		A	→	2
Ç		A		Ç	→	H
Ç		A		O	→	2
Ç		O		A	→	2
Ç		O		Ç	→	2
Ç		O		O	→	2
O		Ç		A	→	2
O		Ç		Ç	→	2
O		Ç		O	→	2
O		A		A	→	2
O		A		Ç	→	1
O		A		O	→	H
O		O		A	→	2
O		O		Ç	→	1
O		O		O	→	H

2. amaç için kuralları yazarken Otoyol 2'nin de akış yoğunluğunu hesaba katabilirdik fakat bunu yaptığımız vakit kurallar çok karmaşıklaşacak ve örnek petri ağı kurmak zorlaşacaktı. O yüzden sadece Yol1, Yol3 ve Otoyol1'in 'sözel' akış yoğunluklarını hesaba kattık.

4.1.3.3. Amaçlanan Petri Ağlarının Tasarımı ve Analizi

4.1.3.1'deki amaçlara uygun olarak 4.1.3.2'deki kurallar kullanılarak biri trafik ışığı ile ilgili diğer uyarı ekranı ile ilgili olmak üzere iki adet petri ağı tasarlanmıştır. Bu petri ağlarının tasarımında ve analizinde ;

- Python 2.3 programlama dili ve bu dilde geliştirilen Petri net analiz programı
- Visual Objects Net ++ *
- PIPE *
- Simulaworks *

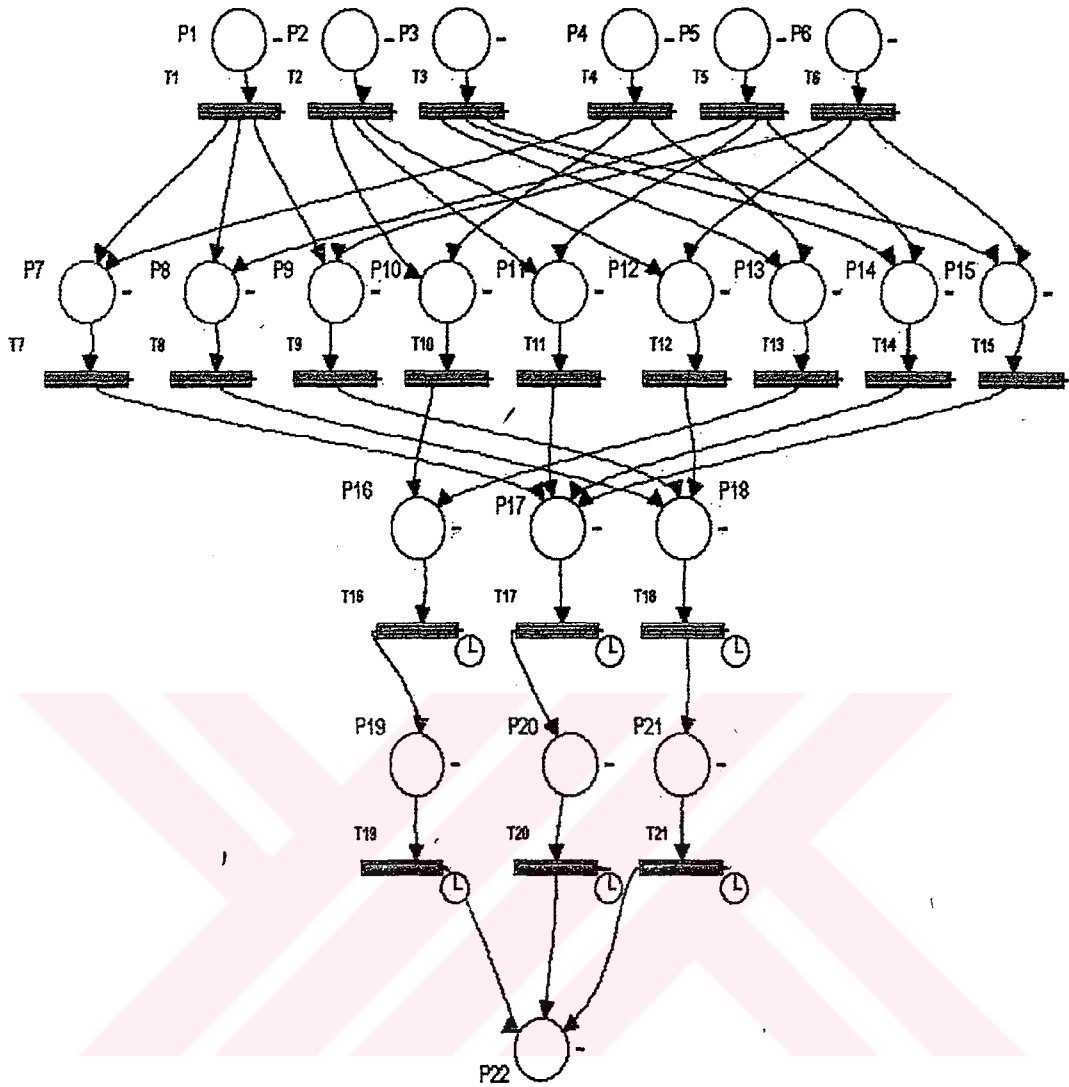
programları kullanılmıştır.

Trafik Işığı Petri Ağı

Tablo 4.1'deki kurallara göre Simulaworks programında hazırlanan Petri Ağı modeli Şekil 4.3'de gösterilmiştir. Bu petri ağında toplam 22 tane (P1,...,P22) yer ve 21 tane geçiş (T1,...,T21) vardır.

Bu petri ağında P1, P2 , P3 yerleri Yol 1'in sırasıyla AZ, ORTA, ÇOK yoğunlukta olduğu değerleri gösteren yerler, P4, P5, P6 'da sırasıyla Yol 3' ün AZ, ORTA, ÇOK yoğunlukta olduğu değerleri gösteren yerlerdir. Tablo 4.1'deki kurallara göre sırasıyla P7, P8, P9, P10, P11, P12, P12, P14, P15 yerleri oluşturulur. Bu yerlerden çıkan okların ağırlığı 2'dir. Yani bu yerler, P1,P2,P3'den birinden bir jeton P4,P5,P6'dan birinden bir jeton olmak üzere iki jeton geldiğinde ilgili geçişlerini ateşleyebilirler. İlgili geçişler ateşlendiğinde keza başlangıç durumuna göre her seferinde sadece tek bir yer iki jeton kazanabileceği için P16, P17, P18 yerlerinden yalnız biri jeton kazanır.

* Bu programlarla ilgili bilgiler ve dosyalar Tez'in EKLER kısmında ve TEZ CD'sinde mevcuttur



Şekil 4.3 Trafik Işığı için Petri Ağı

P16, P17, P18 yerleri trafik ışığının yeşil yanmasının sırasıyla AZ, ORTA ya da ÇOK olarak yanması olaylarına karşılık gelir. P16, P17, P18 yerlerinden birisi (keza her seferinde sadece biri jeton kazanabilir) jeton kazandığında, örneğin başlangıç durumunda sadece P1 ve P3 yerlerinde jeton var ise ilgili geçişlerin ateşlenmesi sonucunda P17 yeri jeton kazandığında, petri ağının zamanlı petri ağı kısmına geçilmiş olur. Burada T16, T17, T18, T19, T20, T21 yerleri zamanlı geçişlerdir yani bu geçişler oklardan kendilerine yeterli miktar jeton geldiğinde anında ateşlenmezler ayarlanan belli bir süre sonrasında otomatik olarak ateşlenirler.

Tablo 4.3 Trafik Işıđı için kurallara göre yerler

	P1 veya P2 veya P3	P4 veya P5 veya P6	P16 veya P17 veya P18
P7→	A	A	O
P8→	A	O	Ç
P9→	A	Ç	Ç
P10→	O	A	A
P11→	O	O	O
P12→	O	Ç	Ç
P13→	Ç	A	A
P14→	Ç	O	O
P15→	Ç	Ç	O

Burada ışık süresini 120 saniyelik bir periyot olarak düşünmekteyiz. Aynı zamanda Yol 1 ve Yol 3'ün akış yoğunluğu değerlerinin her 120 saniyede bir yenilendiğini ve de P17–P21 arasındaki geçişlerdeki gecikmeler haricinde petri ağında hiçbir gecikme olmadığını varsaymaktayız.

Yeşil ışığın ardından sarı ışık olmaksızın direk kırmızı ışığın yandığını ve ilk olarak yeşil ışığın yandığını varsayarsak 120 saniyelik örnek bir periyot (bu periyot otoyol akış yoğunluđuna göre deđiştirilebilir) için aşağıdaki kuralları oluşturabiliriz.

EĐER (120 sn.lik bir periyotta)

Yeşil Işık **AZ** süreli ise → 30 sn yeşil ışık ,ardından 90 sn. kırmızı ışık yanar.

Yeşil Işık **ORTA** süreli ise → 60 sn yeşil ışık, ardından 60 sn. kırmızı ışık yanar.

Yeşil Işık **ÇOK** süreli ise → 90 sn.yeşil ışık, ardından 30 sn. kırmızı ışık yanar.

Tüm bunlara bakarak 360 saniyelik bir periyottaki yeşil ışık durumları örnek başlangıç durumlarına göre Tablo 4.4'de gösterilmiştir.

Tablo 4.4 Örnek başlangıç durumlarına göre I1'in ışık süreleri

T (sn)	Yol 1 Akış Yoğ.	Yol 3 Akış Yoğ.	Yeşil Işığın 'Sözel' Yanma Süresi	Yeşil Işık Süresi (sn)	Kırmızı Işık Süresi (sn)
1-120	A	A	O	60	60
120-270 ⁴	A	O	Ç	90	30
270-360 ⁴	O	O	O	60	60

Simulaworks programında zamanlı geçişlerin zamanını saniye cinsinden değil de geçişlerin ateşlenme hızları olarak verebildiğimiz için yukarıdaki kurallara ve Tablo 4.3'e dayanarak Simulaworks programında hazırlanan TEZ CD'sindeki "isik.xml" dosyasındaki petri ağında zamanlı geçişler ;

30 sn'lik gecikme için 3 birim hızlı

60 sn'lik gecikme için 1,5 birim hızlı

90 sn'lik gecikme için 1 birim hızlı

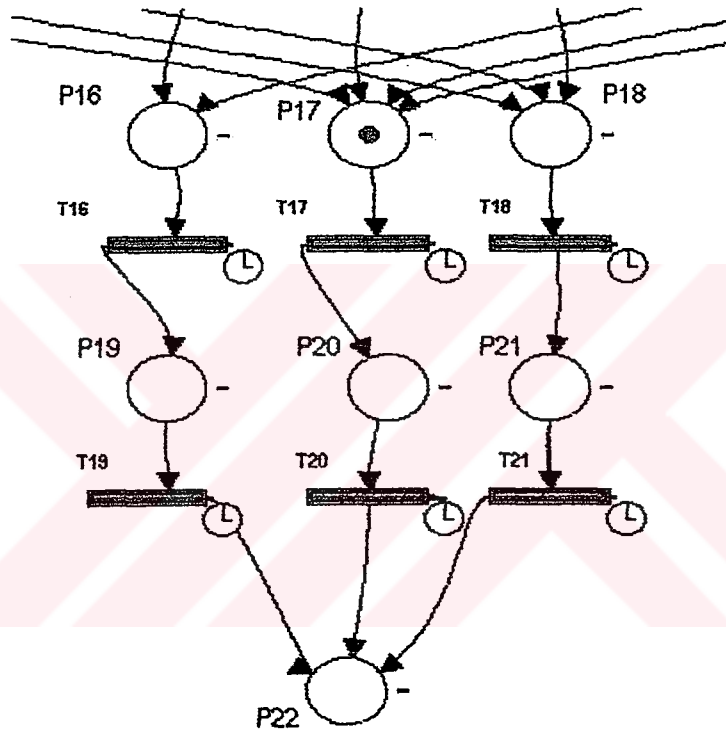
olarak tasarlanmıştır. Buna göre petri ağındaki zamanlı geçişlerin hızları ;

$V(T16) = 3 \text{ br.}$ $V(T17) = 1,5 \text{ br.}$ $V(T18) = 1 \text{ br.}$

$V(T19) = 1 \text{ br.}$ $V(T20) = 1,5 \text{ br.}$ $V(T21) = 3 \text{ br.}$ 'dir.

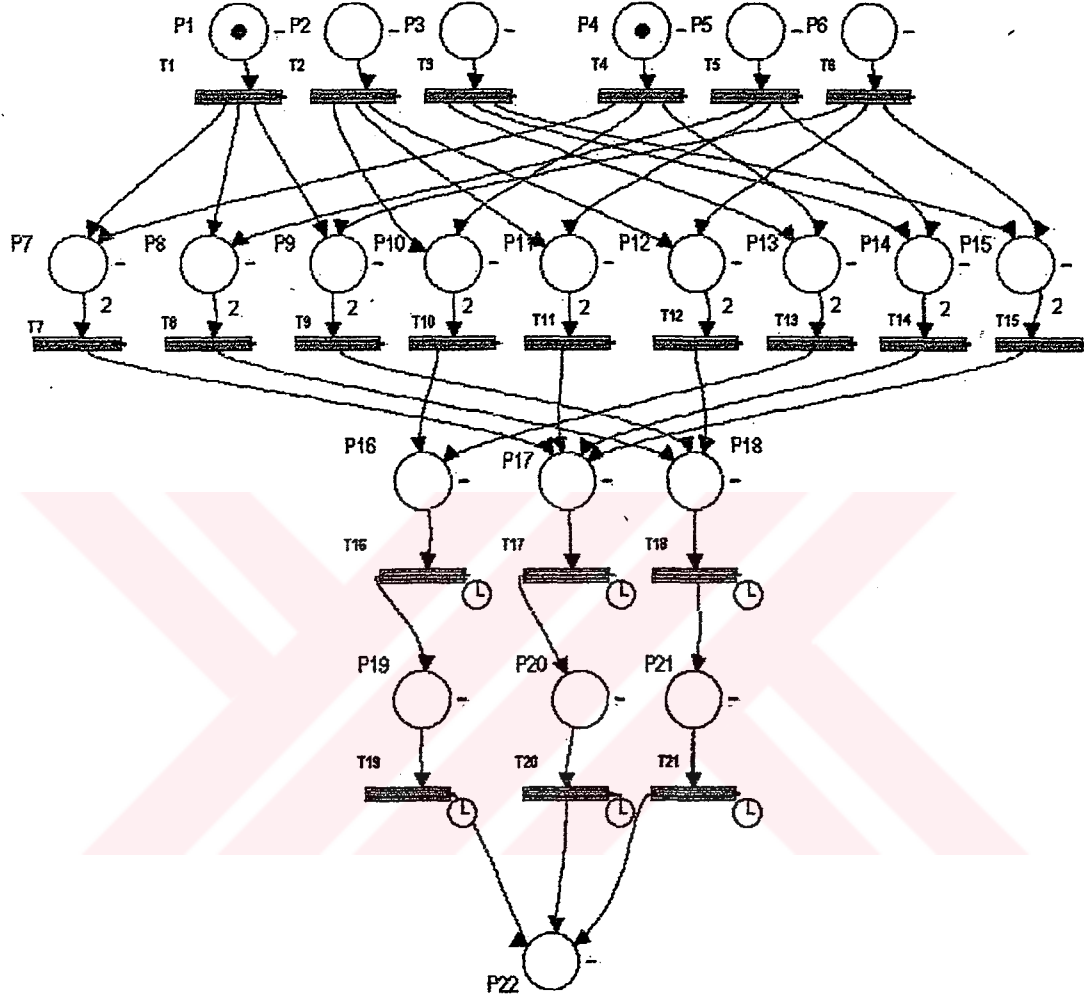
O halde örnek olarak $P1 = A$, $P3=A$ başlangıç durumu için $P17$ yerine jeton geldiğinde $T17$ geçişi başlangıç konumuna gelir 1,5 br hızlıdır yani temsili olarak 60 sn bekler ateşlenir ve $P20$ yeri jeton kazanır. $P20$ jetonu kaybettiğinde $T20$ başlangıç konumuna gelir ve 1,5 br hızlıdır yani temsili olarak 60 sn. bekler ve sonrasında ateşlenir ve $P22$ jeton kazanmış olur. Petri ağı ilk 120 saniyelik periyot için sonlanmış olur. İkinci periyot için Simulaworks'de petri ağını resetleyip yeni başlangıç değerlerini (yerler için jetonları) girmek gerekir. Bu şekilde Tablo 4.4'deki

TEZ CD'sindeki Simulaworks program dosyası "isik.xml"de bu petri ağının başlangıç durumları değiştirilerek (giriş yerlerine jetonlar eklenerek) L1 lambasının ışık süreleri ile ilgili ağdaki değişimler gözlemlenebilir. Örnek olarak Yol1 = A ve Yol3 = A koşulları için Simulaworks programında Şekil 4.5, Şekil 4.6, Şekil 4.7, Şekil 4.8, Şekil 4.9'daki durumlar gözlenmiştir.



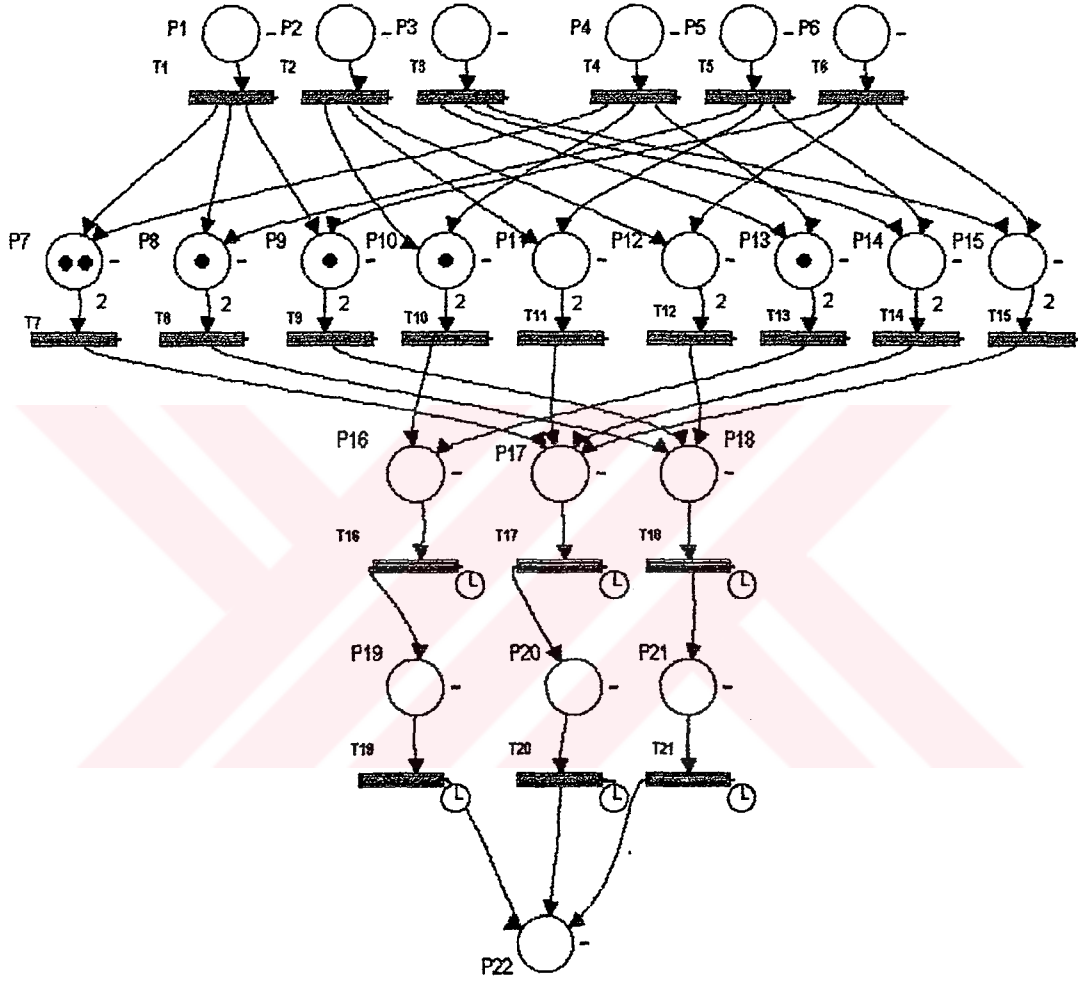
Şekil 4.4 Trafik Işığı için Petri Ağındaki Zamanlı Geçişler

Başlangıç durumunda P1, P4'de birer jeton bulunur yani durum vektörü $X_0=[1\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]$ 'dır (Şekil 4.5).



Şekil 4.5 Trafik Işığı Petri Ağı için başlangıç durumu

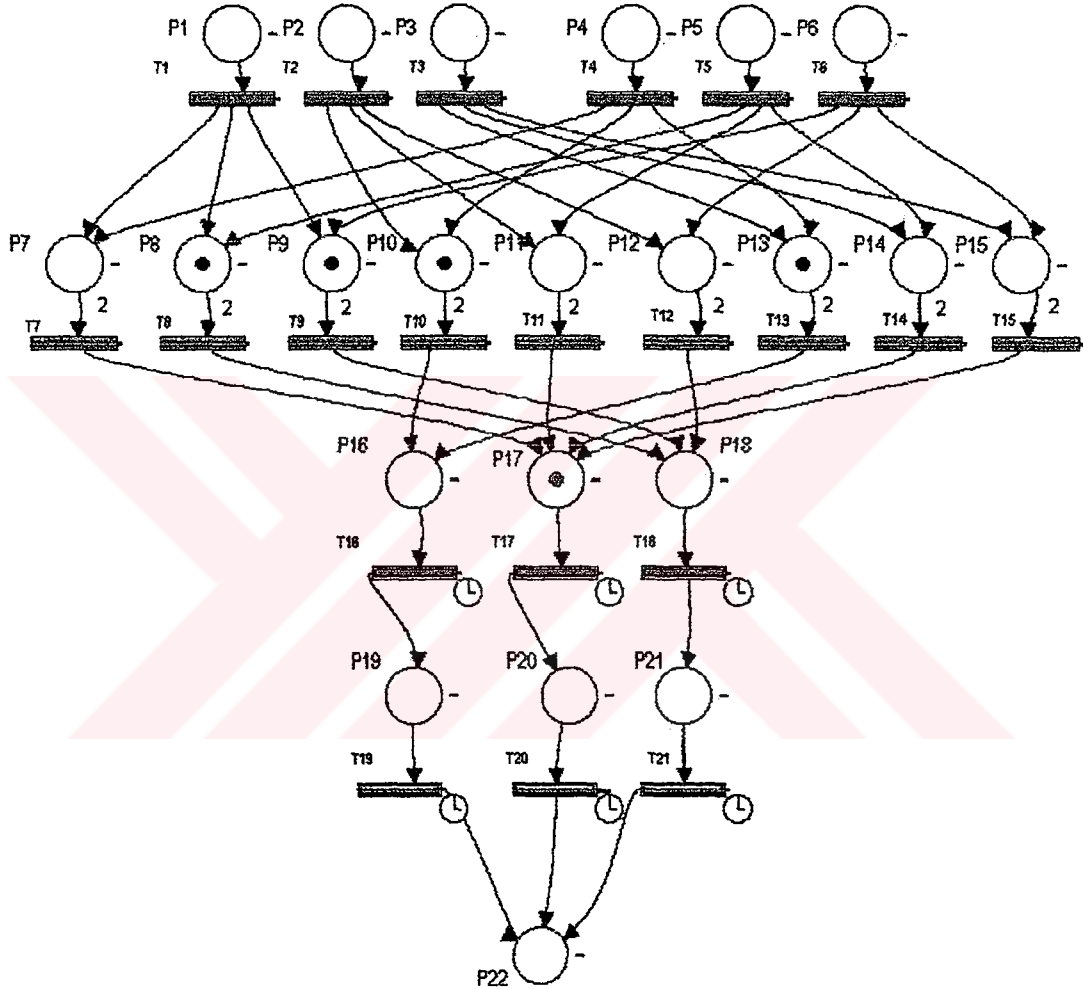
İkinci aşamada T1 ve T4 ateşlenir. Sonrasında P8, P9, P10, P13 birer jeton P7 ise iki jeton kazanır. $X_1 = [0\ 0\ 0\ 0\ 0\ 2\ 1\ 1\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]$ 'dır (Şekil 4.6).



Şekil 4.6 Trafik Işığı Petri Ağı için ikinci aşama

Üçüncü aşamada iki jetona sahip olan P7 jetonunu kaybeder ve T7'nin ateşlenmesini sağlar ve P17 jeton kazanır.

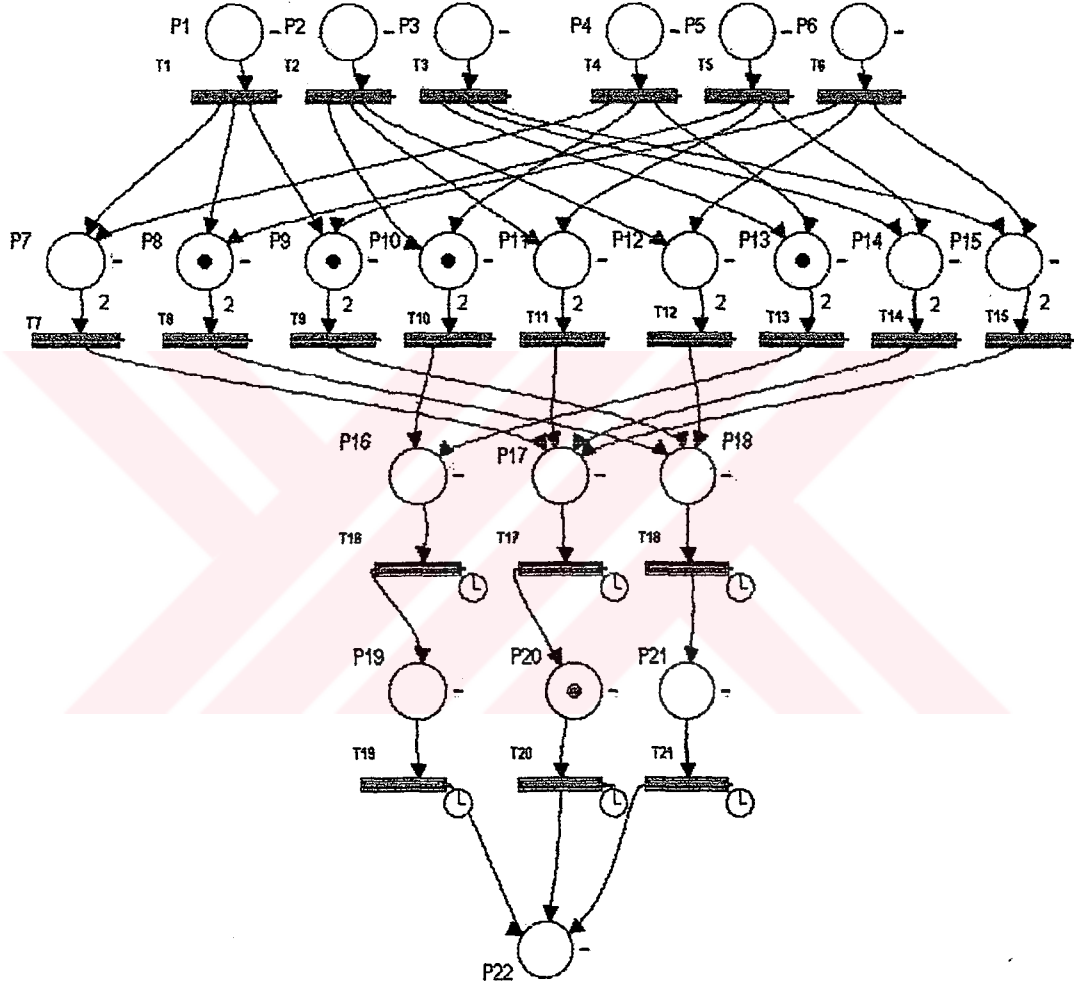
$X_2 = [0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0]^T$ 'dir (Şekil 4.7).



Şekil 4.7 Trafik Işığı Petri Ağı için üçüncü aşama

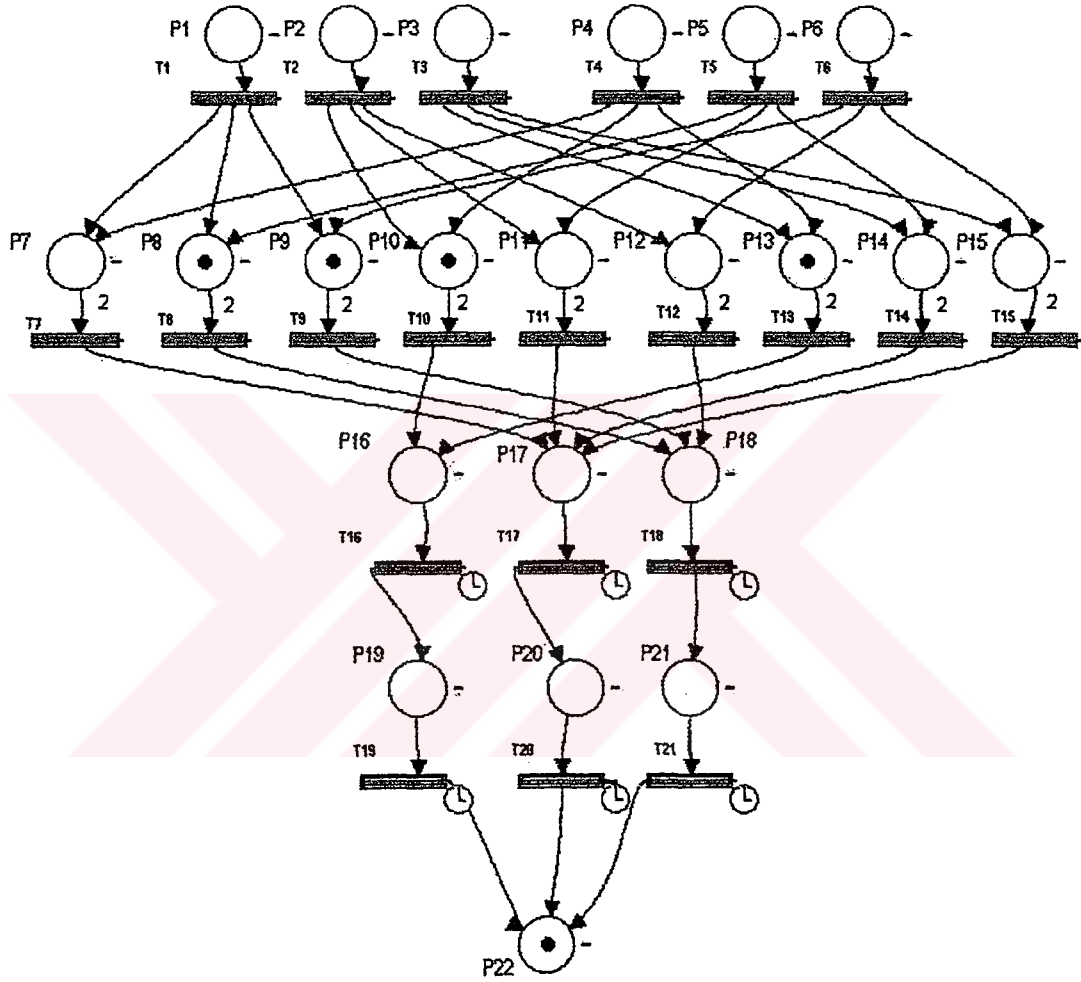
Dördüncü aşamada T17 temsilen 60 sn. gecikmeli olarak (1,5 hızlı olarak) ateşlenir ve P17 jetonunu kaybeder, P20 jeton kazanır.

$X_3 = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0]$ 'dir (Şekil 4.8).



Şekil 4.8 Trafik Işığı Petri Ağı için dördüncü aşama

Son aşamada T20 temsilen 60 sn. gecikmeli olarak (1,5 hızlı olarak) ateşlenir ve P20 jetonunu kaybeder, P22 jeton kazanır ve temsilen 120 saniyelik ışık sureli petri ağı sonlanmış olur. $X_4 = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1]^T$ 'dir (Şekil 4.9).



Şekil 4.9 Trafik Işığı Petri Ağı için son aşama

Şimdi de Simulaworks'de modellenen petri ağının, Python programlama dili ile yazılmış olan programlardan[6] faydalanılarak oluşturulan TEZ CD'sindeki "textanal.py" ve "pnetanal.py" programları yardımıyla analizini aşağıdaki aşamalar ile gerçekleştirebiliriz.

Programın sağladığı analizler :

- Sınırlılık (Boundedness) Analizi
 - o Program, incelenen petri ağı eğer sınırlı ise kaç sınırlı olduğunu bize bildirir(örn : "petri ağı 2 sınırlıdır" diye bildirir). Eğer sınırlı değilse "w sınırlıdır" diye bildirir.
- Sakınım (Conservation) Analizi
 - o Program, kullanıcıdan petri ağındaki yerlerin sayısı kadar katsayıya sahip olan bir katsayı vektörü r ile c sakınım faktörünü girmesini bekler. Sonucunda petri ağının girilen r vektörlü c değerini koruyup korumadığını bildirir.
- Ölü Geçiş (Dead Transition) Analizi
 - o Program, petri ağındaki herhangi bir geçişin ölü geçiş olup olmadığını girilen geçişe göre bildirir.
- Ulaşılabilirlik (Reachability) Analizi
 - o Program, eğer petri ağı sınırlı ise ağıdaki herhangi bir yere ulaşılabilip ulaşılamayacağını girilen yere göre bildirir.
- Canlılık (Liveness) Analizi
 - o Program, eğer petri ağı sınırlı ise herhangi bir yerin L0 ya da L4 canlı olup olmadığını bunlardan birisi değilse L2'den ya da L3'den biri olduğunu bildirir. Fakat L2 mi L3 mü onu bildirmez. Programda kullanılan algoritma henüz o aşamaya kadar geliştirilememiştir.

Programın örnek petri ağlarının yazımı

Programa petri ağlarının tanıtılması için bir kelime işlemci programında (örneğin Windows'da Notepad programında) ağlar python programının kurallarına göre yazılır ve ".py" uzantılı olarak kaydedilir..Daha sonra bu petri ağlarının dosya isimleri "textanal.py" programı çalıştırıldığında bilgisayara girilmesi istenecektir.

Şekil 3.8'deki örnek için Python programında ".py" uzantılı petri ağı dosyası oluşturmak için Notepad'e aşağıdaki formatta yazılması gerekir :

```
p = ['p1', 'p2']
t = ['t1', 't2', 't3']
w = { ('p1', 't1') : 1, ('t1', 'p2') : 1, ('p2', 't2') : 1, ('p1', 't2') : 1, ('p1', 't3') : 1, ('t3', 'p1') : 1 }
```

Burada ;

p = ['p1', 'p2'] → yerler

t = ['t1', 't2', 't3'] → geçişler

w → ağırlık vektörü

w'deki argumanların anlamları –örneğin- ;

('p1', 't1') : 1 → p1 yerinden t1 geçişine 1 ok

('t1', 'p2') : 1 → t1 geçişinden p2 yerine 1 ok var demektir.

Eğer istenirse Sstate vektörü de dosyanın içine yazılır. Sstate vektörü başlangıç durum vektörüdür. Eğer bu dosyanın içine yazılmazsa "textanal.py" programı çalıştırıldığında program kullanıcıdan başlangıç durum vektörünü girmesini ister. Burada –örneğin-

Sstate= (0,1,0)

olarak yukarıdaki formatta yazılabilir.

Vektörler yukarı gösterilen formatlarda yazıldıktan sonra –örneğin- "pnet2.py" olarak kaydedilir.

Programın yüklenmesi

TEZ CD'sindeki Phyton 2.3 set up programı çalıştırılarak Phyton platformu bilgisayara yüklenir. TEZ CD'sindeki "textanal.py", "pnetanal.py", "otoyol.py", "isik.py" ve istenirse örnek bir petri net .py uzantılı dosyası (pnet2.py) Phyton'un yüklendiği dizindeki Phyton klasörünün içindeki "Bin" klasörünün içine kopyalanır. (Bu sayede phyton platformu bu dosyaları farkedebilecektir). Klasörünün içerisine "python.exe" dosyası çalıştırılır. Gelen komut satırına "import textanal" yazılır. Böylelikle petri ağı programı aktif hale geçer.

“isik.py” dosyasında olan Trafik ışığı için Petri ağının Analiz edilme aşamaları Yol1 = A ve Yol3 = A ($X_0(p1)=1$, $X_0(P4)=1$) başlangıç koşulları için komut satırındaki işlemler Windows 98 işletim sistemli bir bilgisayarda aşağıdaki gibi gerçekleştirilir.

Python 2.3 (#46, Jul 29 2003, 18:54:32) [MSC v.1200 32 bit (Intel)] on win32

Type "help", "copyright", "credits" or "license" for more information.

```
>>> import textanal
```

Lutfen Petri agi tanimlamasi iceren (.py uzantili) dosyayi belirtiniz: isik.py

Lutfen Petri aginin baslangic durumunu belirtiniz: (1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)

Lutfen Analiz Gorevini Seciniz:

1. Sinirliligi(Boundedness) inceleme
2. Sakinim (Conservative) inceleme
3. Olu Gecis (Dead Transition) inceleme
4. Ulasilabilirlik (Reachability)inceleme (sadece sinirli Petri agi icin)
5. Canlilik (Liveness) inceleme (sadece sinirli Petri agi icin)
6. iptal

1

Petri ağı 2 sınırlıdır.

Lutfen Analiz Gorevini Seciniz:

1. Sinirliligi(Boundedness) inceleme
2. Sakinim (Conservative) inceleme
3. Olu Gecis (Dead Transition) inceleme

4. Ulasilabilirlik (Reachability) inceleme (sadece sinirli Petri agi icin)
5. Canlilik (Liveness) inceleme (sadece sinirli Petri agi icin)
6. iptal

2

Lutfen katsayi vektoru r yi belirtiniz (format: [int, int, ...]): [2,3,4,5,6,7,7,7,7,7,7,7,6,6,5,6,4,3,2,1]

Lutfen sakinim faktoru c yi belirtiniz (format: int): 7

Petri agi [2, 3, 4, 5, 6, 7, 7, 7, 7, 7, 7, 7, 6, 6, 5, 6, 4, 3, 2, 1] kat sayilari ile olan 7 degerini korumaz

Lutfen Analiz Gorevini Seciniz:

1. Sinirliligi(Boundedness) inceleme
2. Sakinim (Conservative) inceleme
3. Olu Gecis (Dead Transition) inceleme
4. Ulasilabilirlik (Reachability) inceleme (sadece sinirli Petri agi icin)
5. Canlilik (Liveness) inceleme (sadece sinirli Petri agi icin)
6. iptal

3

Lutfen kontrol etmek istediginiz gecisi giriniz (format: 'char'): 't10'
t10 gecisi oludur.

Lutfen Analiz Gorevini Seciniz:

1. Sinirliligi(Boundedness) inceleme
2. Sakinim (Conservative) inceleme
3. Olu Gecis (Dead Transition) inceleme
4. Ulasilabilirlik (Reachability) inceleme (sadece sinirli Petri agi icin)
5. Canlilik (Liveness) inceleme (sadece sinirli Petri agi icin)
6. iptal

4

Lutfen ulasilabilirliğini kontrol etmek istediginiz durumu giriniz (format: (int , int, ...)): (1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1)
(1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1) durumu ulasil amazdir.

Lutfen Analiz Gorevini Seciniz:

1. Sinirliligi(Boundedness) inceleme
2. Sakinim (Conservative) inceleme
3. Olu Gecis (Dead Transition) inceleme
4. Ulasilabilirlik (Reachability)inceleme (sadece sinirli Petri agi icin)
5. Canlilik (Liveness) inceleme (sadece sinirli Petri agi icin)
6. iptal

5

Lutfen canlilikini kontrol etmek istediginiz gecisi giriniz (format: 'char'): 't17'
t17 gecisi en az L1-canlidir, ama L4-canli degildir

Analizin Sonucu :

"isik.py dosyasında kayıtlı olan $X_0=[1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]$ başlangıç durum vektörlü petri ağının yukarıda seçilen analizler sonucunda;

- 2 sınırlı olduğu,
- [2, 3, 4, 5, 6, 7, 7, 7, 7, 7, 7, 7, 6, 6, 5, 6, 4, 3, 2, 1] katsayıları ile olan 7 değerini korumadığı,
- T10 geçişinin ölü olduğu,
- [1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1] durumunun ulaşılabilir olduğu,
- T17 geçişinin en L1 canlı olduğu ama L4 canlı olmadığı

Öğrenilmiştir. Bu örnekte takip edilen yoldan giderek programda farklı geçişler, yerler, durumlar için de bu analizler yapılabilir .

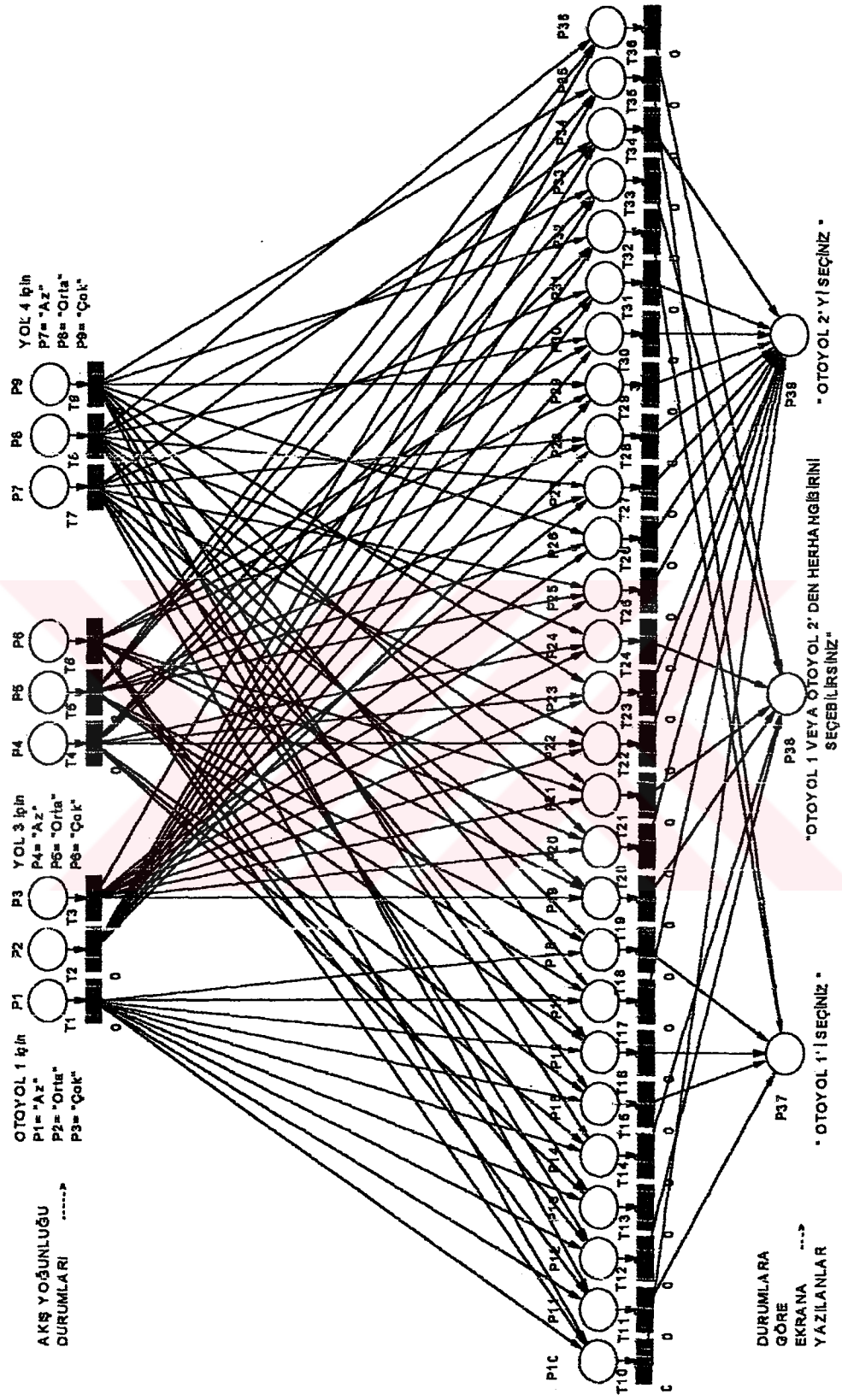
Uyarı Ekranı Petri Ağı

Tablo 4.2'deki kurallara göre Visual Objects Net++ programında hazırlanan Petri Ağı modeli Şekil 4.10'da gösterilmiştir. Bu petri ağında toplam 39 tane (P1,...,P39) yer ve 36 tane geçiş (T1,...,T36) vardır.

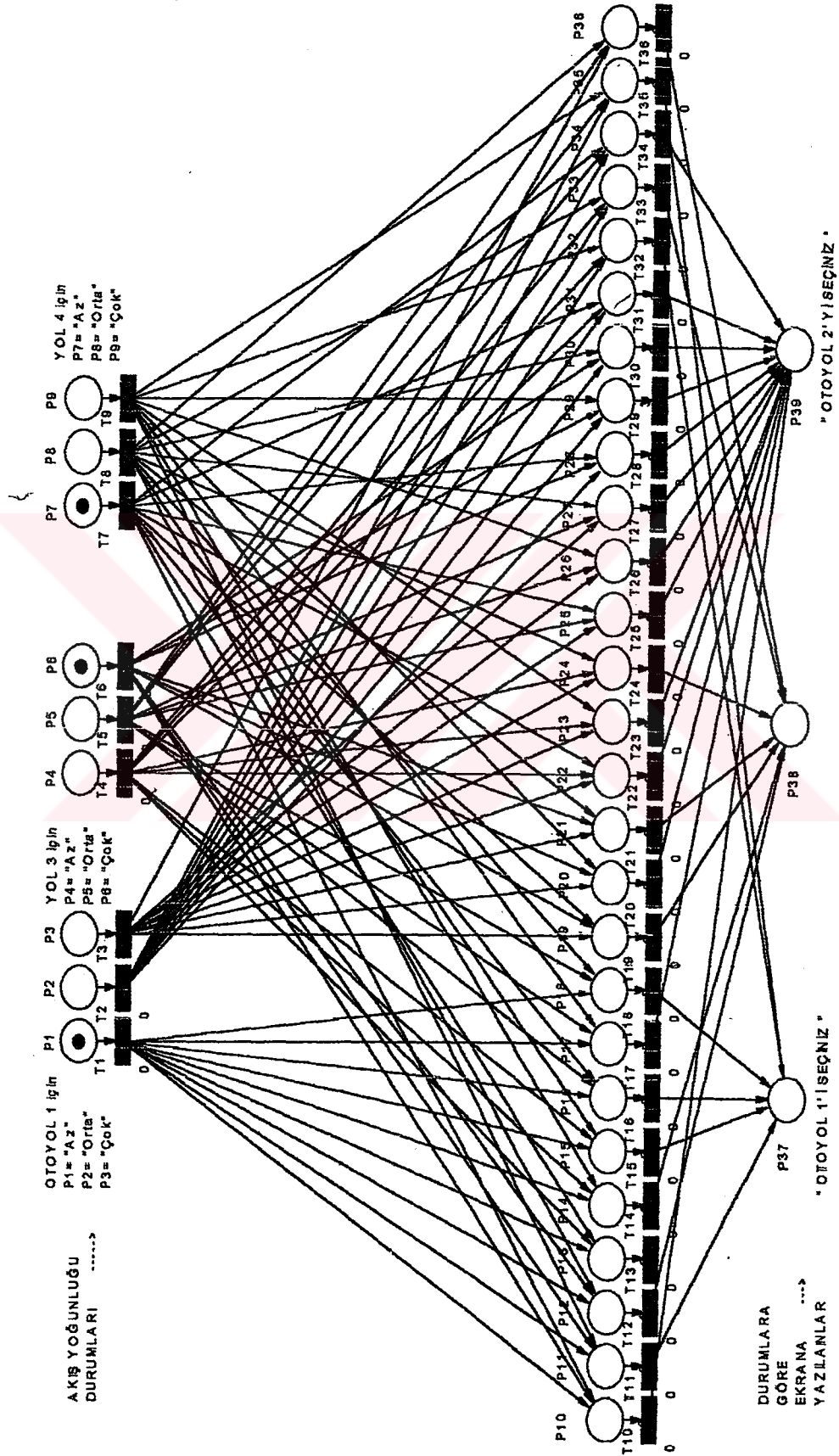
P1, P2 , P3 yerleri Yol 1'in sırasıyla AZ, ORTA, ÇOK yoğunlukta olduğu değerleri gösteren yerler, P4, P5, P6 'da sırasıyla Yol 3' ün AZ, ORTA, ÇOK yoğunlukta olduğu değerleri gösteren yerler, P7, P8, P9 'da sırasıyla Yol 4' ün AZ, ORTA, ÇOK yoğunlukta olduğu değerleri gösteren yerlerdir. Tablo 4.2'deki kurallara göre sırasıyla P10, P11, P12,..., P36 yerleri oluşturulur. Bu yerlerden çıkan okların ağırlığı 3'dür. Yani bu yerler, P1,P2,P3'den birinden bir jeton P4,P5,P6'dan birinden bir jeton P7,P8,P9'dan birinden bir jeton olmak üzere 3 jeton geldiğinde ilgili geçişlerini ateşleyebilirler. İlgili geçişler ateşlendiğinde keza başlangıç durumuna göre her seferinde sadece tek bir yer 3 jeton kazanabileceği için P10,..., P36 yerlerinden her seferinde yalnız biri üç jeton kazanır. Bu üç jetonu kazanan yer kazandığı gibi jetonları kaybederek ilgili geçişini ateşler. Bunun sonrasında P37, P38, P39 yerlerinden biri jeton kazanır ve petri ağının işleyişi ilk başlangıç durumlarına göre tamamlanmış olur. Uyarı ekranına P37 jeton kazanmışsa " OTOYOL 1'İ SEÇİNİZ !"

P38 jeton kazanmışsa " OTOYOL 1 ve OTOYOL 2'DEN HERHANGİBİRİNİ SEÇEBİLİRSİNİZ", P39 jeton kazanmışsa "OTOYOL 2' Yİ SEÇİNİZ !" yazdırılır.

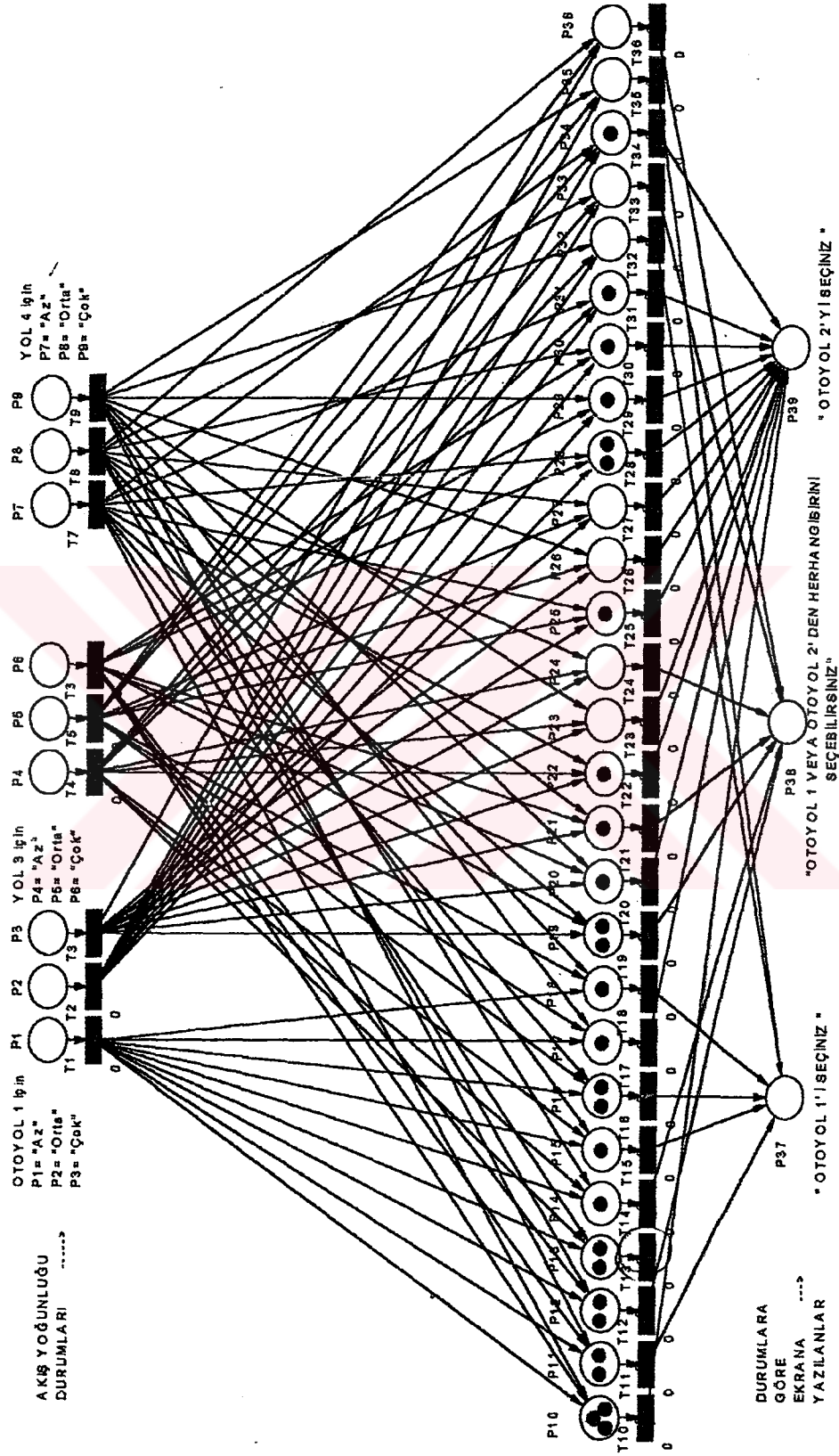
TEZ CD'sindeki Visual Objects Net++ program dosyası "otoyol.pn"de bu petri ağının başlangıç durumları değiştirilerek (giriş yerlerine jetonlar eklenerek) ağdaki değişimler gözlemlenebilir. Örnek olarak $Y_1 = A$ VE $Y_3 = Ç$ VE $Y_4 = A$ koşulları için Visual Objects Net++ programında Şekil 4.11, Şekil 4.12, Şekil 4.13'deki durumlar gözlenmiştir.



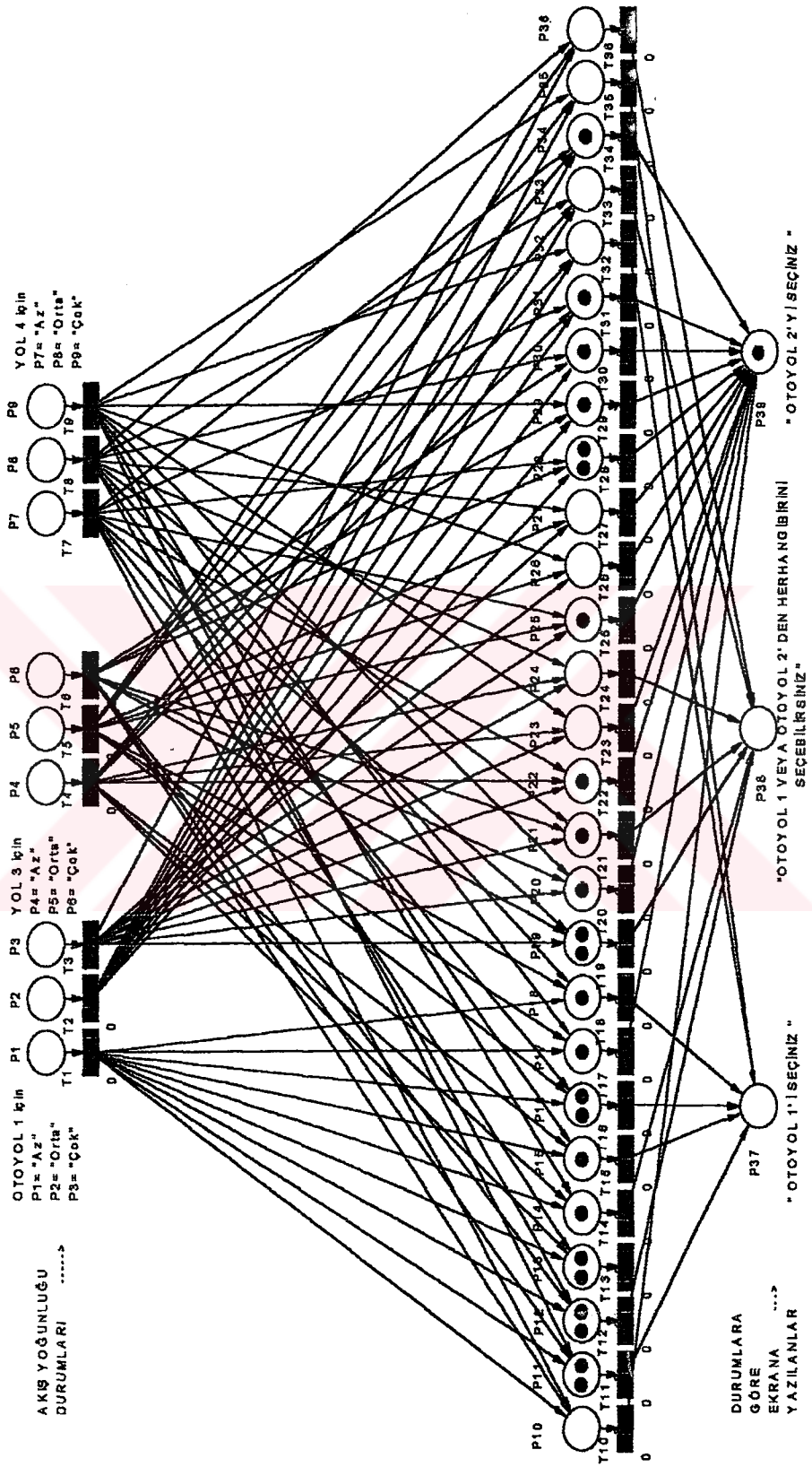
Şekil 4.10 Uyan Ekranı ile ilgili Petri ağı



Şekil 4.11 Uyarı Ekranı ile ilgili Petri ağı başlangıç durumu



Şekil 4.12 Uyarı Ekranı ile ilgili Petri ağı 2. aşama



Şekil 4.13 Uyarı Ekranı ile ilgili Petri ağı son aşama

"otoyo1.py" dosyasında olan Uyarı Ekranı için petri ağının analiz edilme aşamaları
 $Y_1 = A$ VE $Y_3 = Ç$ VE $Y_4 = A$ başlangıç koşulları için komut satırındaki işlemler
Windows 98 işletim sistemli bir bilgisayarda aşağıdaki gibi gerçekleştirilir

Lutfen Analiz Gorevini Seciniz:

1. Sinirliligi(Boundedness) inceleme
2. Sakinim (Conservative) inceleme
3. Olu Gecis (Dead Transition) inceleme
4. Ulasilabilirlik (Reachability)inceleme (sadece sinirli Petri agi icin)
5. Canlilik (Liveness) inceleme (sadece sinirli Petri agi icin)
6. iptal

Lutfen Analiz Gorevini Seciniz:

1. Sinirliligi(Boundedness) inceleme
2. Sakinim (Conservative) inceleme
3. Olu Gecis (Dead Transition) inceleme
4. Ulasilabilirlik (Reachability)inceleme (sadece sinirli Petri agi icin)
5. Canlilik (Liveness) inceleme (sadece sinirli Petri agi icin)
6. iptal

48

1. Sinirliligi(Boundedness) inceleme
2. Sakinim (Conservative) inceleme
3. Olu Gecis (Dead Transition) inceleme
4. Ulasilabilirlik (Reachability)inceleme (sadece sinirli Petri agi icin)
5. Canlilik (Liveness) inceleme (sadece sinirli Petri agi icin)
6. iptal

5

Lutfen canliligini kontrol etmek istediginiz gecisi giriniz (format: 'char'): 't
36'

t36 gecisi L0-canlıdır

Analizin Sonucu :

"otoyol.py" dosyasında kayıtlı olan $X_0=[1,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]$ başlangıç durum vektörlü petri ağının yukarıda seçilen analizler sonucunda;

- 3 sınırlı olduğu,

- [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1] katsayıları ile olan 1 değerini korumadığı,
- T20 geçişinin ölü olduğu,
- [0, 1, 2, 2, 1, 1, 2, 1, 1, 2, 1, 1, 1, 0, 0, 1, 0, 0, 2, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 1] durumunun ulaşamazdır olduğu,
- T36 geçişinin L0 canlı olduğu,

öğrenilmiştir. Bu örnekte takip edilen yoldan giderek programda farklı geçişler, yerler, durumlar için de bu analizler yapılabilir .

4.2. Akıllı Araba ve Akıllı Yollar İçin Petri Ağı Uygulaması

4.2.1 Giriş

Otoyol sıklığı, kentte yaşayanlar için çekilmez bir sıkıntıdır. Bu sıklığı gidermenin yollarından birisi de Akıllı Araç / Otoyol Sistemidir (Intelligent Vehicle / Highway System (IVHS)). Bununla birlikte, “akıllı “ kavramının yer aldığı değişik bölümler mevcuttur. Bunlar :

- Görev : Otomatik hale getirilmesi gereken sürücü görevleri ve otomasyonun derecesi.
- Mimari : Bu görevlerin, kontrol görevlerine ayrıştırılması ve bu görevlerin IVHS alt sistemlerine atanması.
- Dizayn : Araçlar ile otoyol arasındaki “akıllılık” bölümü ve şu anki teknolojilerin bu yapıyı nasıl gerçekleştireceği..
- Evrim : Sistemin gelişimi ve yayılmasının zamanı
- Değer : Farklı IHVS tekliflerinin verimi, maliyeti ve yararları

4.2.2 IHVS Görevleri

Tablo 4.5, geziye çıkmış bulunan bir araba sürücüsü tarafından yapılan bir sıra altı karardan oluşmaktadır. Bu kararlar, gezi öncesi, gezi esnasında ve gezi sonrası aşamalarına bölünmüştür. Ayrıca her karara uygun IVHS amaçları ve uygun şekilde yerine getirildiği takdirde daha somut görevler bu amaçları desteklemektedir. Son sütun, bilgi sağlama, tavsiyede bulunma , kararın direk kontrolünü gerçekleştirme ya da ücretlendirme ile dürtüleri değiştirme gibi çeşitli stratejiler kullanılarak görevlerin gerçekleştirilebileceğini göstermektedir. [7]

Bu altı karar zinciri bir sürücünün verdiği daha geniş kararlar kümesinin çekirdeğini oluşturmaktadır. Örneğin, gezi esnasındaki kararlar kümesine, geçiş ücreti ödemek için durma, benzin için durma, telefonu kullanma , radyo dinleme, yolcularla konuşma vs. gibi kararlar da girebilir

Tablo 4.5’de aynı zamanda alınan kararların daha yüksek kapasite ve güvenliği sağlama için nasıl geliştirilebileceğini göstermektedir. İlk olarak, iyi bir güzergah seçimi seyahat zamanını azaltabilir. Bu seçim, otoyola yabancı olan sürücüler için

faydalı olan haritalar ile "offline" bilgi ile ve de değişen trafik koşullarına göre güzergah belirlenmesini sağlayan trafik gecikmelerindeki (kaza ya da yinelenen sıkışıklık yüzünden olan) gelişmeler hakkındaki "online" bilgi ile geliştirilebilir.

Tablo 4.5 Sürücü Kararları ve IVHS Görevleri

Aşama	Sürücü Kararı	IVHS Amacı	IVHS Görevi	Strateji
Gezi öncesi	- Gezi üretimi, modal seçimi, vs.	-Daha verimli kaynak kullanımı	Talep Değişimi	B, T, Ü
	- Güzergah seçimi	-Seyahat zamanını kısaltma	Güzergah rehberliği ve akış kontrolü	B, T, K
Gezi esnasında	Yol planlaması	Düzenli trafik	Sıkışıklık kontrolü	B, T, K
	Manevra	Güvenliği, akışı artırma	Araç koordinasyonu	B, K
	Düzenleme	Güvenliği, akışı artırma	Uygun boşluk bulma, sürme vs.	B, K
Gezi sonunda	Park etme vs.	Değer ekleme	Verimli kullanım	B, T, K, Ü

T = Tavsiye K = Kontrol B = Bilgi Ü = Ücretlendirme

Kanıtlar göstermektedir ki güzergah seçimini ve yol planlama kararlarını iyileştirmekle %15'lik bir kapasite arttırımı sağlanmaktadır. [7]

Manevralar, sürücülerin , otoyola giriş ve çıkışlarını de içeren şerit değiştirmelerinden oluşmaktadır. Manevralar, komşu araçların hareket koordinatlarına ihtiyaç duyar. Uygun olmayan koordinatlar sıkışıklığına ve kazalara neden olur. Bugünkü otomatik olmayan trafik sisteminde koordinasyon sinyalizasyon (dönüş sinyalleri ve fren ışıkları) ve sosyal anlaşma (şerit değiştirmeye niyet etmiş bir sürücüye bitişik şeritten bir boşluk sağlama) ve trafik işaretleri ile sağlanmaktadır. Bir kısmi-otomatik sistem, çarpışma uyarı sinyali ya da geçici

olarak sürücünden önce harekete geçen çarpışmadan sakınma sistemi sağlayarak manevraları geliştirebilir.

Düzenleme, bir sürücünün aynı şeritte kalırken aracın hızını ayarlaması ile ilgilidir. Kararlı durum koşullarında bu hız $v(d)$ öndeki araç ile olan mesafe d 'nin artan bir fonksiyonudur. Araçlar / şerit / dakika cinsinden $\phi(d)$ akışı metre / dakika cinsinden hızın ve metre başına araç sayısının bir ürünüdür.

$$\phi(d) = v(d) \times \frac{1}{d+s} \quad (4.1)$$

Burada s , araç uzunluğudur. Sürücünün " tepki fonksiyonu" $v(d)$ otoyolun fiziksel koşullarına (şerit genişliği, aydınlanma, yüzey koşulları vs.) bağlı olmakla birlikte deneysel çalışmalar maksimum yaklaşık 40 araç/ şerit /dk değerini önermektedir. [7]

4.2.3. Tam Otomatikleştirilmiş Sistem

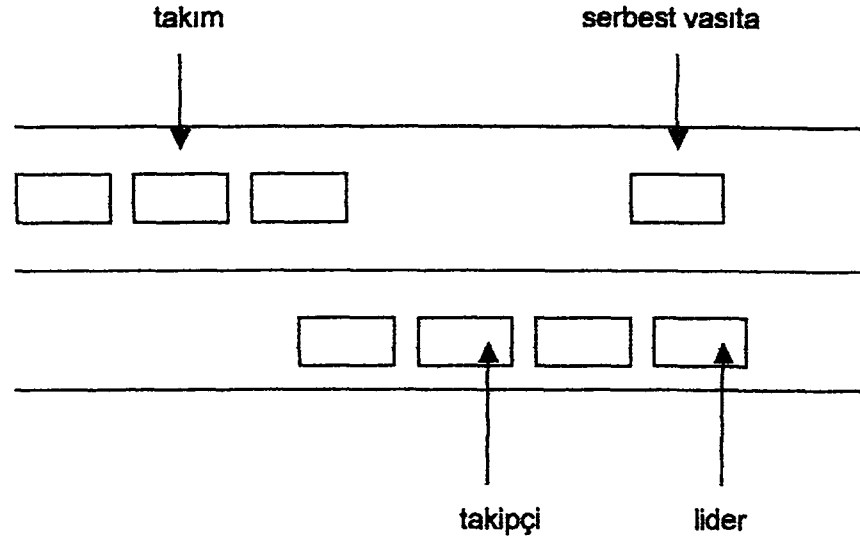
4.2.3.1 Takım Kavramı

Kapasiteyi arttırıcı etkenlerden birisi de Şekil 4.14'de gösterildiği gibi trafikte sıkışık düzendeki takımlar organize etmektir. Takımlar içi mesafe d , takımlar arası mesafe D , araç uzunluğu s , ve sabit durum hızı v metre/dk, eğer trafik n adet araba takımından oluşmuşsa aşağıdaki kapasiteyi ya da akışı vermektedir :

$$\phi(d) = v \times \frac{n}{ns + (n-1)d + D} \text{ araç / şerit /dk} \quad (4.2)$$

$s = 5$ m ve $v = 1,200$ m/dk ya da 72 km /h için bazı değerler Tablo 4.2'de gösterilmiştir.

$\phi = 35$ değerini veren $n = 1$ durumu bugünün kapasite sınırlarını göstermektedir. [2]
 $n = 20$ durumu yapılabılır bir durumdur. Kapasiteyi dörde katlamaktadır. 4.2 ifadesi özellikle araç büyüklüğü ve hızı gibi kapasiteye etki eden diğer dizayn değişkenlerini de ele alır. Birincisi güvenlikle ilgilidir. İkincisi de takımların nasıl oluşturulacağı ve bir takım sisteminde araçların nasıl manevra yapacakları ile ilgilidir. [7]



Şekil 4.14 Takım Kavramı

Tablo 4.6 Takımlarla Kapasite

n	d	D	ϕ
1	—	30	35
5	2	60	64
15	2	60	105
20	1	60	133
∞	1	—	200

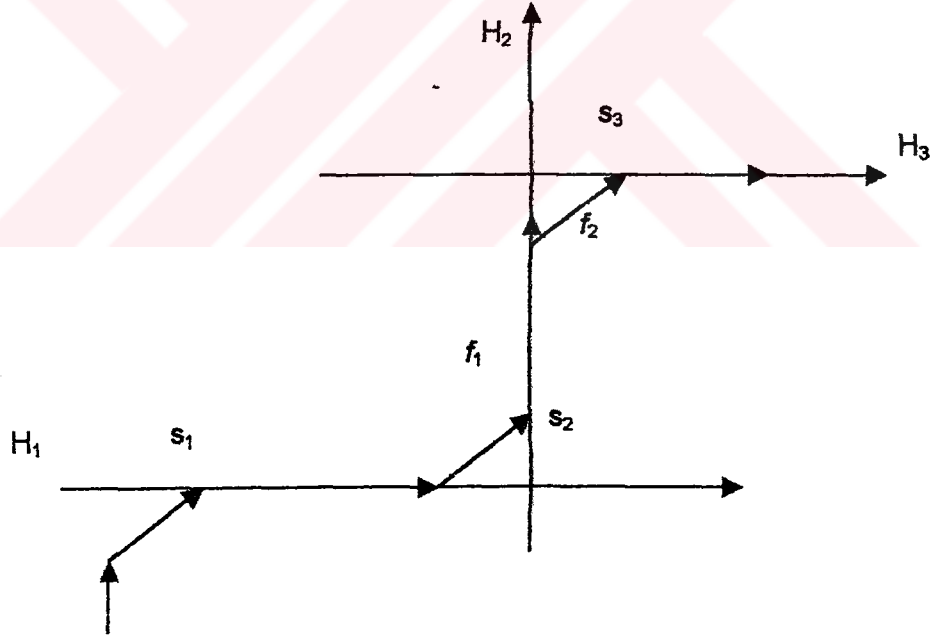
Bir insan sürücüsünün tepki için gecikme süresi 0.25 ile 1.2 saniye arasındadır ki bu bir sürücüyü kesin güvenli kılmaz. Araç otomatik kontrol altında olmalı ki aniden hızını düşürebilsin. [7]

4.2.3.2 Bir IVHS Senaryosu

Birbirine bağılı otoyollardan oluşan otomatikleştirilmiş bir otoyol ağı hayal edelim. Uygun şekilde donatılmış araçlar, bu otoyol ağına çeşitli "geçit"lerden girdiğini ve çıktığını ve bu ağ boyunca otoyol ve araçlar arasında bulunan bir otoritenin kontrolünde seyahat ettiğini düşünelim. Bu otoyol ağı, çeşitli homojen olmayan ağları içeren daha büyük taşıma sistemlerin içerisinde gömülü durumda bulunabilir. Örneğin, bir araç otomatikleştirilmiş ağı terkederken otomatikleştirilmemiş bir şehir merkezi ağına girebilir.

Bir geçitten girdiğinde araç, varacağı noktayı bildirir. IVHS sistemi, ağ içinden araca bir güzergah atarak bildiriye cevaplar. Bu güzergah (route) R aşağıdaki şekildedir :

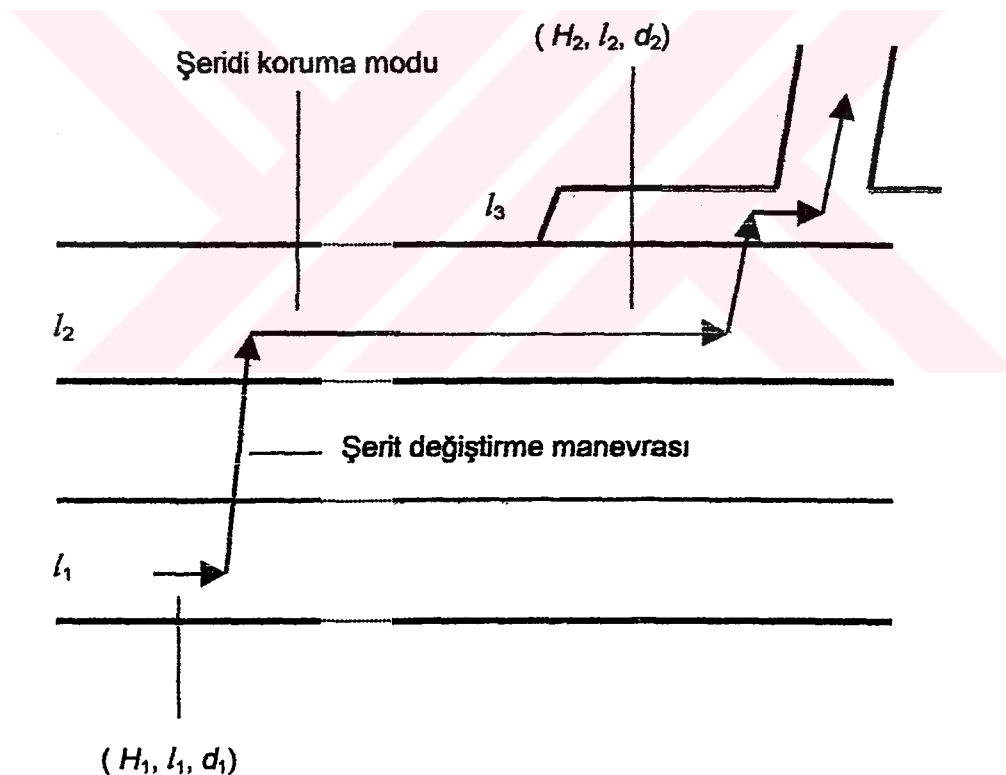
$$R = (H_1, s_1, f_1), (H_2, s_2, f_2).. \quad (4.3)$$



Şekil 4.15 Bir güzergahın bir dizi kısımdan meydana gelmesi

Güzergah bir dizi kısımdan meydana gelecek şekilde üretilmiştir. Güzergahın ilk kısmı otoyol (highway) H_1 , başlangıç geçidi s_1 , bitiş geçidi f_1 ; ikinci kısım, s_1

Bir aracın sürekli olarak hangi bölgede seyahat ettiğini algıladığını varsaymaktayız. Bir 'bölge' bir H 'nin otoyol adı, I 'nin şerit numarası d 'nin şerit bölge numarası olduğu bir (H, I, d) üçlemidir. Bir bölge yaklaşık 100m uzunluğundadır. Aracın otoyol H 'ye bölge (H_1, I_1, d_1) 'deki bir s geçidinden girdiğini ve (güzergahına bağlı olarak) f geçidinden çıkacağını bildirdiğini düşünelim. Karşılık olarak sistem araca bir yol (I_2, d_2, I_3) atar. Buradaki anlam, araç I_2 şeridine geçmeli d_2 kısmına kadar seyahat etmeli ve sonra kendisini f geçidine ulaştıracak olan I_3 şeridine girmelidir (Şekil 4.16).

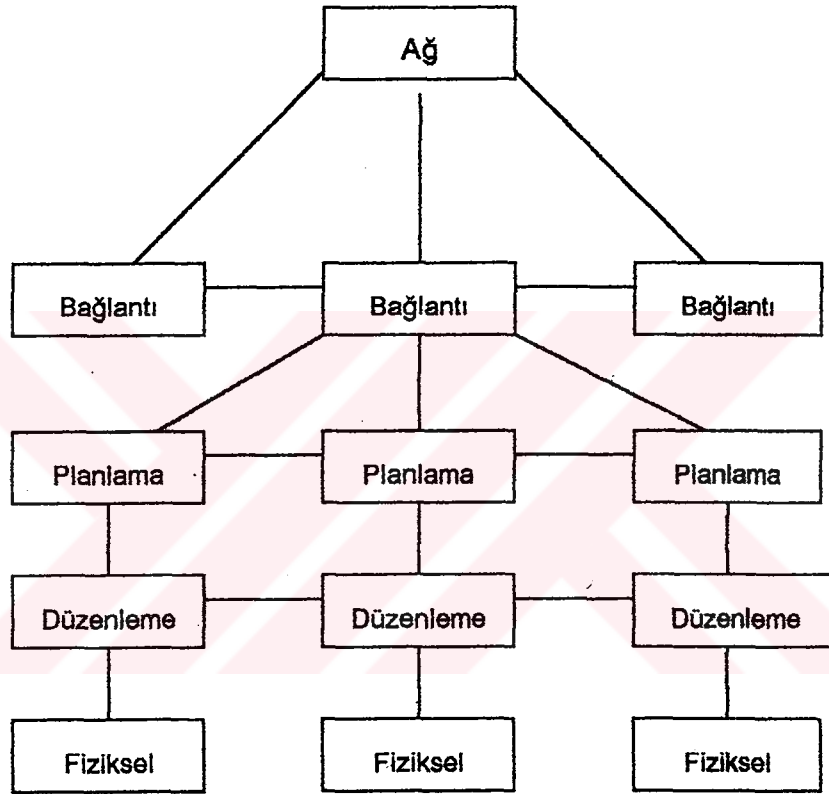


Kendisine bir *yol* atandıktan sonra araç muhakkak bu *yola* yakın gerçek zamanda bir *plan* geliştirmeli ve bu plana uygun bir *hedefi* gerçekleştirmelidir [7].

4.2.3.3 Kontrol Sistemi Yapısı

Şekil 4.17 dört katmanlı bir yapıyı göstermektedir. Yukardan başlayarak katmanlar ağ, bağlantı, planlama ve düzenleme diye adlandırılmaktadır.

Tam otomatikleştirilmiş ağ için tek bir ağ kontrolörü vardır. Bu kontrolörün görevi sisteme giren tüm araçlara bir güzergah atamaktır.



Şekil 4.17 Kontrol Sistem Yapısı

Her otoyolun uzun kısımlar boyunca bir bağlantı kademesi kontrolörü vardır. Görevleri her araç için trafiğini her şerit boyunca dengeleyecek şekilde bir yol ve sıkışıklığı engelleyici, düzgün bir akışı sağlamak ve olaylara adapte olacak şekilde bir hedef hızı atamaktır.

Her araç için bir planlama kontrolörü ve düzenleme kontrolü bulunur. Görevleri sırasıyla : atanan yola yakın bir plan çizmek ve aracın plana uygun olan hedefini gerçekleştirmektir.

Buradaki kontrolör dizaynı çoğu bilgi ve kontrol otoritesini bireysel araçlara dağıtmakta (özellikle düzenleme ve planlama kontrolörleri araç içerisinde, bağlantı ve ağ katman kontrolörleri de yol kenarındadır) ve komşu araçların hareketini koordine ederek belirsizlikleri azaltmaktadır.

Dizaynda ana fikir kontrolü dağıtmak ve araçların hareketlerini sınırlamak ve koordine etmek ve böylece kontrol görevini basitleştirmek. İki adet sınırlayıcı küme bulunmaktadır. Bunlardan birincisinin içerdikleri;

- 1) İlk sıra şerit değiştirmelerde aracın l_1 şeridinden l_2 şeridine geçerken her seferinde yalnız bir şerit değiştirmesi.
- 2) l_2 şeridinden d_2 kısmına kadar bir şeridi koruma modu
- 3) Şerit l_2 'den l_3 'e hareketi sağlayan terminal şerit değiştirme sırası. (Şekil 4.16)

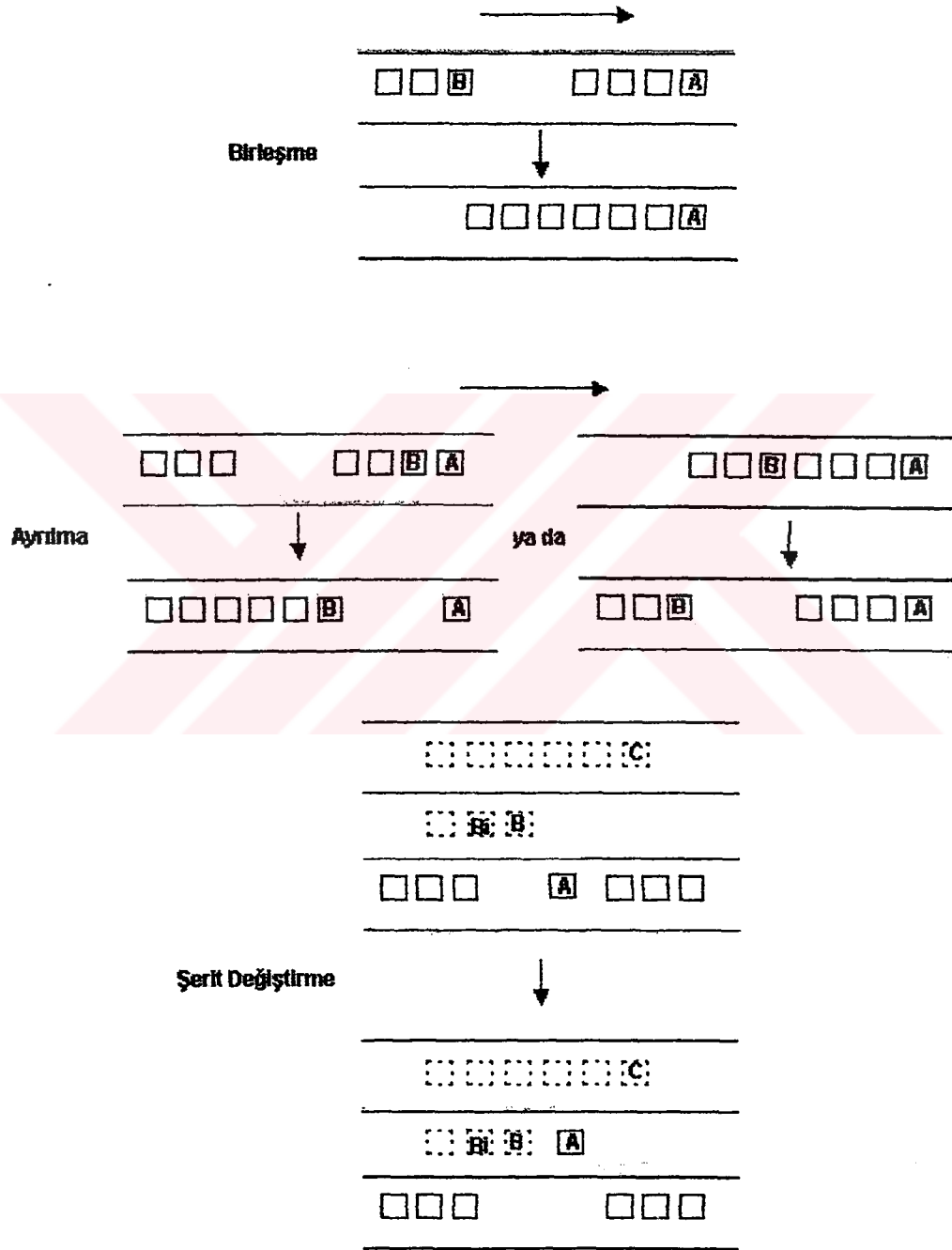
Ek bir sınırlama ise sadece serbest araçlar şerit değiştirebilir olmasıdır (Bir serbest araç tek arabalık takımdır; bir takımın en öndeki aracına lider; diğerlerine takipçiler denir.)

Böylece takım oluşumu ve ayrılması sadece şeridi koruma modunda oluşabilir. Bunun için iki manevraya ihtiyaç vardır : bir birleşme manevrasına, iki takım bir takım oluşturmak için birleşirler; bir ayrılma manevrasında bir takım ikiye ayrılır. Eğer bir takımdaki bir araç bir şerit değişikliği yapmak isterse önce serbest araç haline gelmelidir. Bu, bir ayrılma manevrasını gerektirir (eğer takım içerisinde en öndeki ya da en sondaki araç ise) ya da iki ayrılma manevrası gerektirir. Bu manevralar, Şekil 4.17'de gösterilmiştir. [7]

Bağlantı katmanı kontrolörü otoyolun her kısmına bir hedef takım büyüklüğü N ve hedef hızı v atar. Liderler bu hedeflere , güvenliği koruyarak mümkün olduğu kadar uymak zorundadırlar. İkinci sınırlama kümesi güvenlik ile ilgilidir. Bu kümeye göre :

- 1) Sadece liderler (ve serbest araçlar) manevraları başlatabilir ve takipçiler her zaman takım düzenini korurlar.
- 2) Bir lider her defasında en çok bir adet manevra gerçekleştirir.
- 3) Bir lider, her manevrayı güvenlik için komşu takımların liderleri ile birlikte koordine eder.
- 4) Sadece anlaşmaya varıldığında lider manevrayı gerçekleştirir.

En az iki araçlık takımlar, sadece bir birleşme ya da ayrılma durumunda bir araya gelebildikleri için, bu kararlar bir lider tarafından yerine getirilmelidir. Takipçiler sadece takip eder. Eğer bir takipçi bir serbest araç olmak isterse (şerit değiştirmek için) öncelikle liderine gereken ayrılma manevrasını yapması için talepte bulunması gerekir.



Şekil 4.18 Üç manevra

4.2.4. IVHS'in Planlama Katmanı için Petri Ağı Uygulaması

Şimdiye kadar anlatılmış olan Akıllı Araç / Otoyol Sisteminde Planlama katmanında Petri Ağları uygulaması ile şu üç görev yerine getirilecektir.

- 1) Atanmış planını gerçekleştirmek için hangi manevraya teşebbüs edileceğine karar vermek
- 2) Güvenliği sağlamak için komşu araçlarla beraber manevrayı koordine etmek
- 3) Manevraya karşılık düşen hedefi gerçekleştirmek için düzenleme katmanını denetlemek

Bu görevleri yerine getirebilmesi için planlama katmanı, *planlama_durumu* dediğimiz bir bilgiye ihtiyacı vardır.

Planlama_durumu = (*N*, *v*, *l*, *d*, *arac_kimligi*, *takim_kimligi*, *arac_buyuklugu*, *takim_yeri*, *mesgul*)

N, *v* bağlantı katmanınca atanan hedef takım büyüklüğü ve hızıdır, *l* ve *d*, aracın bulunduğu (alıcılar tarafından saptanan) otoyol şeridi ve bölgesidir. *Arac_kimligi* ve *takim_kimligi* tanımlayıcılardır, *takim_buyuklugu* o anki takımın büyüklüğü, *takim_yeri* aracın takım içerisindeki o anki yeri (*takim_yeri*=1 lider demektir) ve *mesgul* aracın bir manevrada ise aktif durumda olan aksi taktirde pasif duruma geçen bir bayraktır. [7]

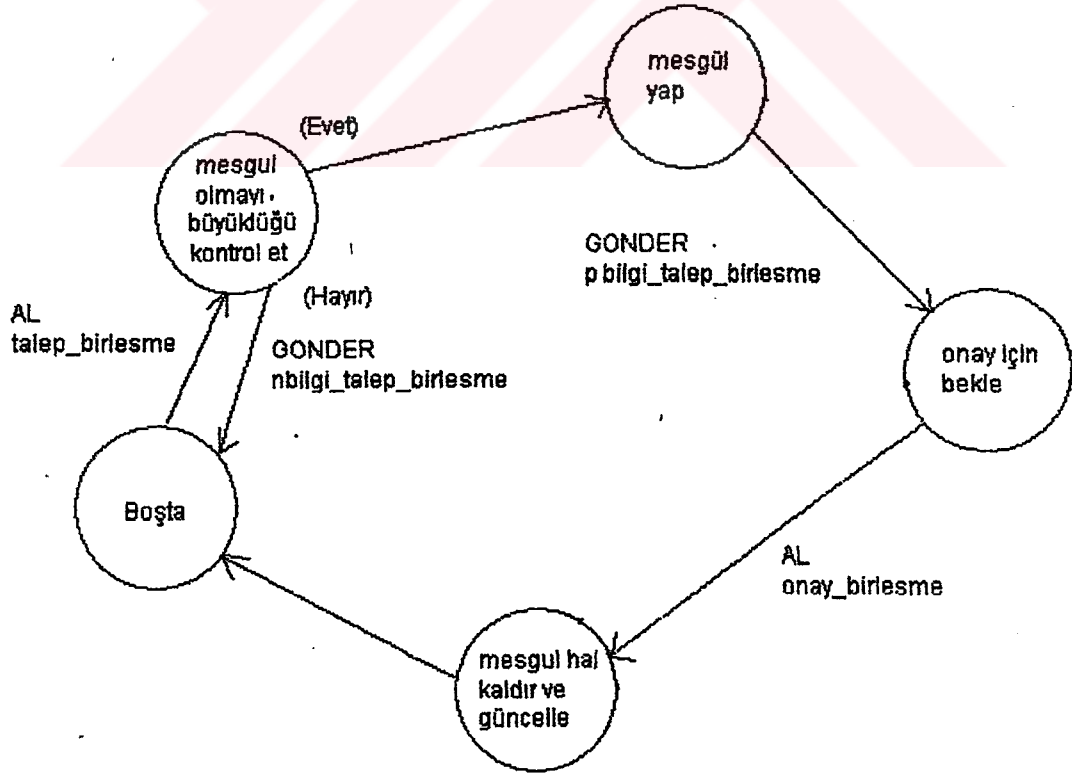
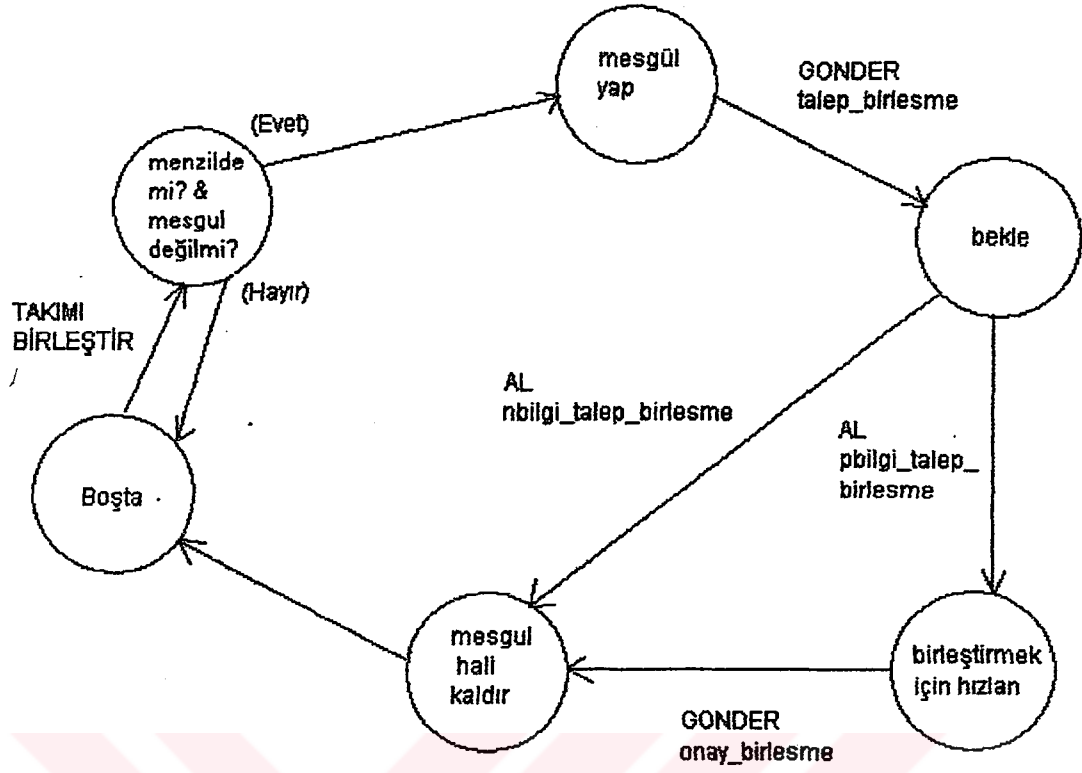
Burada gerçekleştirilecek petri ağı uygulamasında ana sistemin ufak bir modeli oluşturulacaktır. Ana petri ağı sisteminde en üstte YM "Yol Monitörü" adlı bir katman bulunmaktadır. Bu monitör, aracın bulunduğu konumu atanan yolla kıyaslar , bir manevra seçer ve sonra DENET denetleme yapısında manevrayı gerçekleştirmesi için siparişte bulunur. Eğer manevra şerit değiştirme ise ve eğer araç bir serbest araç değilse DENET , bir ya da iki ayırma manevrası yapması için SBAL (serbest araç ol) yapısı ile kontağa geçer. Komşu araçların planlama katmanları ile koordinasyon her üç manevrada da gereklidir ve bunlar alt katmandaki on yapıdan biri ile yapılır. Görevleri haberleşme protokollerine benzer olduğu için Protokol dediğimiz mesaj alışverişlerinde bulunarak aralarında koordinasyonu sağlayan bu yapıların görevleri aşağıda verilmiştir :

Şekil 4.18'de gösterilen birleşme manevrasını düşündüğümüzde bu manevra iki araç arasında bir protokolu gerektirir – arkadaki araç B manevranın gerçekleşmesi için komut verir ve öndeki A aracı ona karşılık verir. B'nin birleşme protokol yapısının

ismi BPKbirles (P → Protokol, K→Komut) . A'nın yapısının ismi APCbirles (C→ Cevap) dir. Tabii ki her aracın kontrolöründe iki yapıdan da bulunmalıdır çünkü ya komut vermek ya da komuta cevap vermek gerekebilir. Şekil 4.6'daki ayrılma durumunda, bir araç kendisini şu dört rolden birinde bulur : bir ayrılma komutu veren ya da ayrılma komutuna cevap veren (APKaynı, APCaynı) bir *lider* olarak; bir ayrılma komutu veren ya da ayrılma komutuna cevap veren (BPKaynı, BPCaynı) bir *takipçi* olarak . Son olarak bir şerit değişikliğinde bir araç dört rolden birinde olabilir: şerit değiştirme manevrası komutu veren (APKdegis) araç A olarak ya da komuta cevap veren yan şeritten bulunan (BPCdegis) bir lider olarak, yan şeritteki bir takipçi (BLPCdegis) olarak ya da şeridin uzak noktasında bulunan (CPCdegis) bir lider olarak. Bu şekilde toplam olarak 10 adet protokol yapısı bulunmaktadır [7].

Şekil 4.19'da bir çift durum yapısı bulunmaktadır. Üstteki yapı birleşme komutu veren B aracında ve alttaki ise talebe cevap veren A aracındadır. İki yapı arasındaki bir mesaj almaya ve göndermeye karşılık düşen geçişler beraber çiftleştirilmiştir. Böylece, örneğin, B'deki 'GONDER talep_birlesme' etiketli geçiş A yapısındaki 'AL talep_birlesme' etiketli geçiş ile aynı anda meydana gelmektedir. Benzer şekilde A'daki 'GONDER pbilgi_birlesme' (pbilgi = pozitif bilgilendirme , nbilgi = negatif bilgilendirme) ve B'deki 'AL pbilgi_birlesme'.

Yapıların çalışması şu şekilde gerçekleşir. Varsayalım ki iki yapı da boşta durumdalar. B yapısının TAKIMI BİRLEŞTİR geçişini yaptığını (Bu aynı zamanda B'nin DENET yapısındaki bir geçişle çiftlenecektir) farzedelim. Sonra yapı, B'nin ön mesafe algılayıcısının menzilinde olan ve birleşme yapılabilecek bir aracın olup olmadığını saptandığı ve meşgul bayrağının pasif durumda olup olmadığını (aracın bir manevraya yönlendirilmemiş durumda olması beklenir) denetlendiği 'menzilde ve mesgul olmayan bir konumda mı kontrol et' durumuna geçer. Her iki sorguya da cevap hayır ise yapı boşta durumuna döner. Eğer cevapların her ikisi de evet ise , B meşgul bayrağını kaldırır ve 'GONDER talep_birlesme' geçişini ateşler, burada, bir birleşme talebini mesafe algılayıcısının yerini tespit ettiği takımın liderine, A'ya gönderir. B daha sonra bekleme durumuna geçer. Eğer şimdi A'dan bir negatif bilgilendirme (AL nbilgi_birlesme) alırsa B mesgul bayrağını pasif duruma geçirir ve boşta duruma geri döner. Eğer pozitif bilgilendirme alırsa (AL pbilgi_birlesme) , hızlanması ve A'nın takımı ile birleşmesini sağlayan düzenleme katmanına sipariş veren bir geçişe ilerler. Düzenleme katmanı birleşme işlemini tamamladığında B, A'ya (GONDER onay_birlesme) bir onay birleşme mesajı gönderir, mesgul bayrağını pasif hale getirir ve boşta duruma geri döner.



Şekil 4.19 Birleşme için Durum Yapıları

A, B'den *talep_birleşme*'yi aldığı anda mesgul olup olmadığını ve birleşmiş durumdaki takım büyüklüğünün hedef büyüklüğün altında olup olmadığını, $takım_buyuklugu(A) + takım_buyuklugu(B) < N$ 'i kontrol eder. Her iki sorgulama için de cevap 'hayır' ise B'ye negatif bilgilendirme gönderir. Eğer her iki cevap da 'evet' ise A kendi mesgul bayrağını(şimdi bir manevraya kilitlenmiştir) aktif hale getirir, B'ye pozitif bilgilendirme gönderir ve *onay için bekle* durumuna geçer. B'den *onay_birleşme* 'yi aldığı zaman, *planlama_durumunu* günceller, mesgul durumunu pasif hale getirir ve boşta çalışmaya geri döner. Birleşme manevrası A ve B'nin her ikisi de boşta duruma dönerek son bulur.

Yukarıda bahsi geçen 'Birleşme Manevrası' nı model alan bir Petri Ağı uygulaması PIPE adlı simülasyon programı ile gerçekleştirilmiştir.

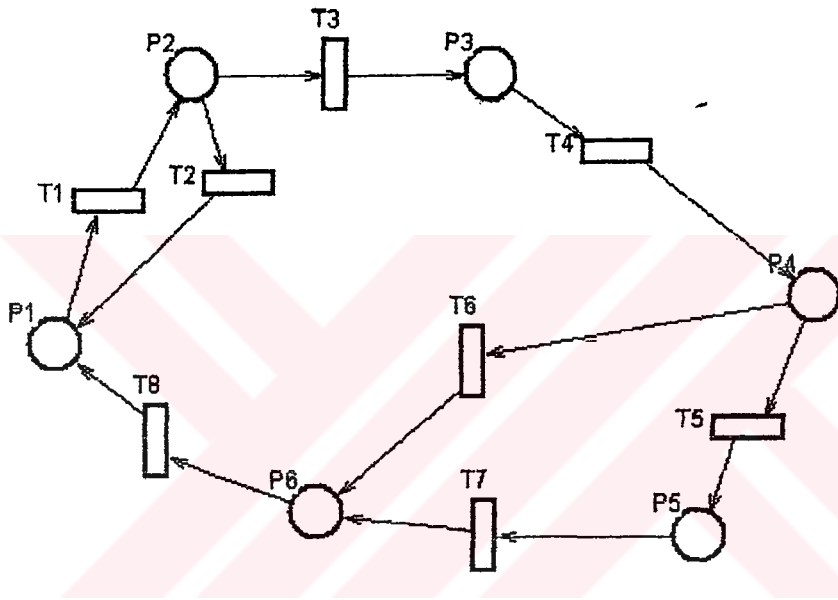
PIPE simülasyon programını kullanmamızın sebebi bu programda diğer Petri ağı simülasyon programlarında sık karşılaşılmayan, petri ağındaki hangi geçişin ateşleneceğini kullanıcının seçimine bırakmasıdır. Kullanıcı, sistem işler iken adım adım hangi geçişin ateşleneceğine (elindeki veriler vasıtasıyla) karar verebilir.

Şekil 4.20'deki petri ağı incelendiğinde ;

Bu Petri Ağı, B aracının öndeki bir A aracı ile birleşme haberleşmesi (Şekil 4.19'daki ilk durum) ile ilgilidir. Bu sistem normalde algılayıcılardan gelen bilgiler vasıtasıyla otomatik olarak işler. Fakat biz bu sistemde kullanıcının algılayıcı bilgilerine sahip olduğunu varsayıyoruz ve çakışan geçişlerden hangisinin ateşleneceğine kullanıcının karar vermesini düşünüyoruz (Bir yerden aynı anda iki geçiş ateşlenemez ve zaten bu ağıın kurulma mantığına aykındır.). Bu yüzden PIPE programında kullanıcının birbiriyle çakışan hangi geçişin ateşleneceğine (T3 veya T2, T5 veya T6) elindeki örnek verilere dayanarak karar verir.

Şekil 4.19'da P1 yerinde bir jeton bulunduğu vakit yani B aracı boşta durumunda iken kullanıcı T1 geçişini ateşlenmesine olanak sağlar. Yani TAKIMI BİRLEŞTİR komutu başlatılır. T1 geçişi sayesinde jeton P2 yerine ulaşır. P2'ye jeton ulaşması, sistemin B aracının önünde, menziline giren meşgul olmayan bir A aracının varlığını kontrol edilmesi demektir. Eğer böyle bir A aracı yoksa T2 ateşlenir P2 jeton kaybeder P1 jeton kazanır yani B aracı boşta hale geri döner. Eğer böyle bir A aracı varsa T3 geçişi ateşlenir (PIPE'da bu ateşlemeleri kullanıcı elindeki veriye göre yapar), P2 jetonunu kaybeder, P3 jeton kazanır. Yani B aracı kendisinin meşgul hale geçtiğini belirten meşgul bayrağını aktif hale getirir. Ardından T4 geçişi ateşlenir yani B aracı A aracına "Talep_birleşme" komutunu gönderir, P3 jeton kaybeder P4 jeton

kazanır. Yani B aracı gönderdiği bilgiye cevap bekleme durumuna geçer. Eğer beklenen cevap pozitif ise T5 ateşlenir negatif ise T6 ateşlenir. T5 ateşlenirse jeton P5 konumuna geçer ve B aracı birleşmek için hızlanır. Birleşme tamamlandığında T7 ateşlenir ve P6 yeri bir jeton kazanır yani Meşgul hal kaldırılır. Sonrasında T8 ateşlenir ve P1 jeton kazanır tekrar boşta hale geçilir. Eğer beklenen cevap negatif ise yani T6 ateşlenirse birleşme için hızlanmadan P6 direk jeton kazanır yani B aracının mesgul hali kaldırılması işlemi yapılarak T8 geçişi ile jeton P1'e ulaşır. B aracılık baştaki boşta durumuna geri döner.



P1	-1	1	0	0	0	0	0	1
P2	1	-1	-1	0	0	0	0	0
P3	0	0	1	-1	0	0	0	0
P4	0	0	0	1	-1	-1	0	0
P5	0	0	0	0	1	0	-1	0
P6	0	0	0	0	0	1	1	-1

Şekil 4.20 B aracı için petri ağı ve sistem matrisi A

Şekil 4.20'deki Petri ağındaki Yerler(durumlar) :

P1 : Boşta Durumu

P2 : Menzilde aracın olup olmadığının ve Meşgul bayrağının aktif olup olmadığının sorgulandığı durum.

- P3 :Meşgul bayrağını aktif hale gelme durumu
P4 : Bekleme durumu
P5 : .Birleşmek için hızlanma durumu
P6 : Meşgul bayrağını pasif hale getirme durumu

Geçişler :

- T1 : Takımı birleştir komutunun gönderildiği geçiş
T2 : Aracın menzilde olmadığı ya da mesgul bayrağının aktif olduğu zaman ateşlenen geçiş
T3 : Aracın menzilde olduğu ve mesgul bayrağının pasif olduğu zaman ateşlenen geçiş
T4 : GONDER talep_birlesme komutunun gönderildiği geçiş.
T5 : AL pbilgi_talep_birleşme komutunun gönderildiği geçiş.
T6:: AL npbilgi_talep_birleşme komutunun gönderildiği geçiş.
T7 : GONDER onay_birlesme komutunun gönderildiği geçiş.
T8 : Boşta hale geçiş.

Aşağıda PIPE programında kullanıcının çıkışan geçişlerden hangisini ateşleyeceğine karar vermesi için gereken bazı örnek algılayıcı bilgileri verilmiştir. Bu verilere göre kullanıcı PIPE programında simule edilen bu petri ağını TEZ CD'sindeki "arac_b.xml" dosyasında inceleyebilir.

Tablo 4.7'de PIPE programında modellenen ve "arac_b.xml" dosyasında bulunan yukarıdaki petri ağı için B aracının A ile örnek birleşme denemeleri verilmiştir. Bu tabloda B ile A aracı arasındaki mesafeye , A aracının boşta olup olmamasına , birleşme öncesi B aracının talebine verdiği cevaba göre kullanıcının hangi geçişi ateşleyeceğine karar verebilir. PIPE programında da bu seçimlerinin nasıl gerçekleştiğini jetonların hareketini gözlemleyebilir.

Tablo 4.7'de görüldüğü gibi eğer B ile A aracı arasındaki mesafe 200m.'nin üstünde ise ya da A aracı mesgul durumda ise B aracı boşta çalışmaya geri döner yani T2 mi ateşlenmeli T3 mü sorusu cevabını bulur ve T2 ateşlenir. Eğer mesafe 200m.'nin altında ve A aracı boşta durumunda (meşgul değilse) ise T3 ateşlenir. T3'ün ardından T4'ün ateşlenmesinin sonrasında eğer A'nın B'ye olan cevabı negatif ise T6 ateşlenir pozitif ise T5 ateşlenir. Sonraki ateşlemeler, aralarında çıkışma

olmadığı için rastgele ya da otomatik halde ateşlenebilir (kullanıcının verilere dayanarak verdiği seçime ihtiyaç duyulmaz).

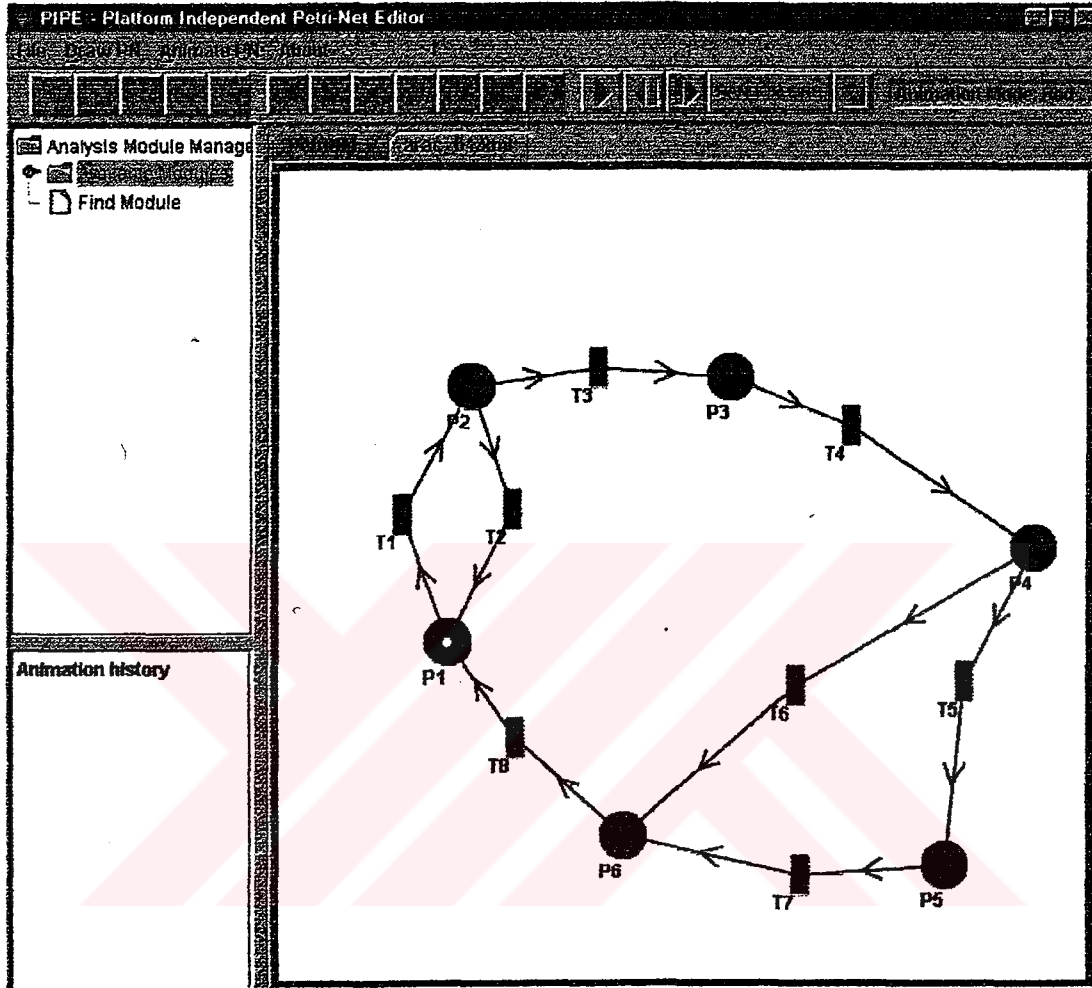
Tablo 4.7 B'nin Birleşme Denemeleri Tablosu

B'nin Birleşme Denemesi	B ile A arasındaki Mesafe (m)	A'nın Durumu	A'nın Cevabı	Geçişlerin Ateşlenme Sırası
1.	130	Meşgul	YOK	T1,T2
2.	270	Boşta	YOK	T1,T2
3.	180	Boşta	Negatif	T1,T3,T4,T6,T8
4.	190	Boşta	Pozitif	T1,T3,T4,T5,T7,T8

* Birleşme manevrası için gereken B aracı ile A aracı arasındaki minimum mesafe'nin 200 m olduğu varsayılmıştır.

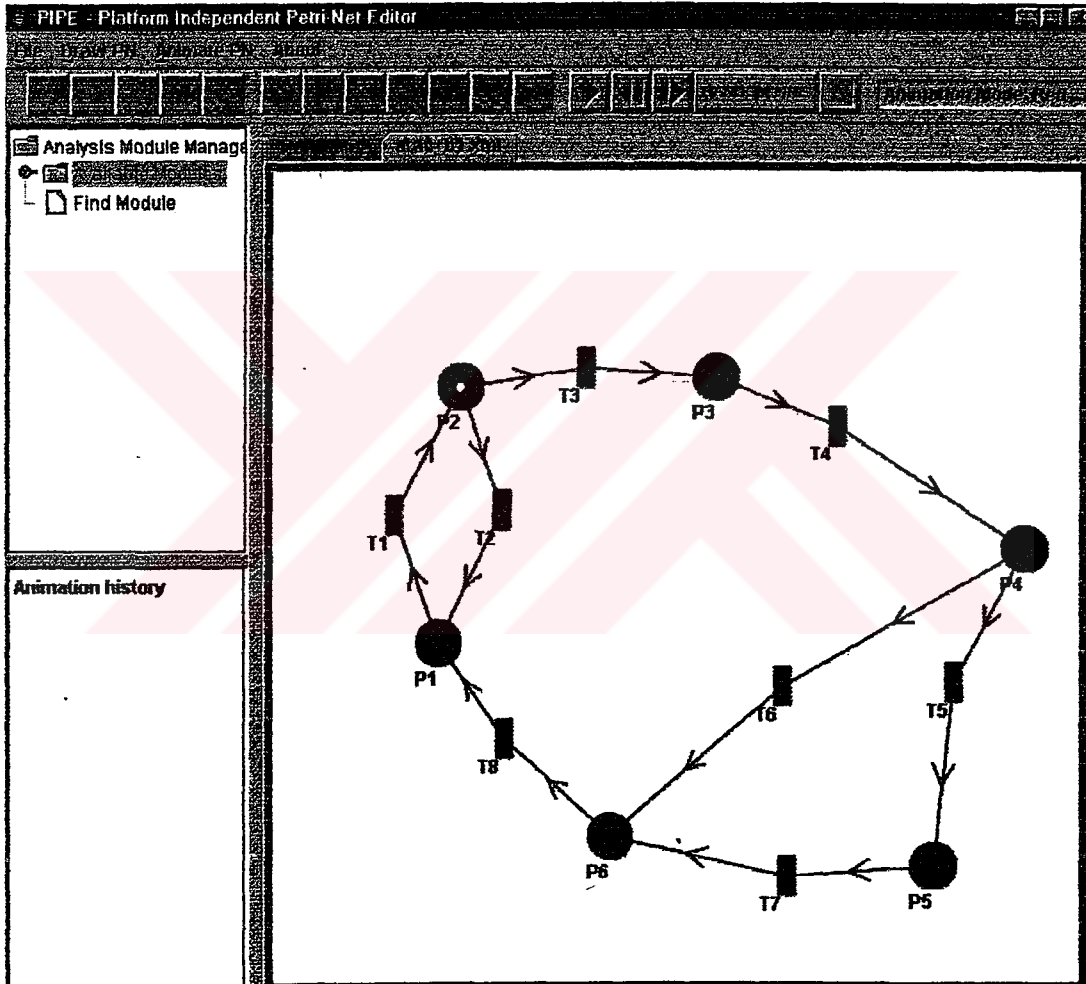
PIPE programındaki "arac_b.xml" dosyasında Tablo 4.5'deki 4.birleşme manevrası için yapılan "run-time" işlemler kısaca şöyledir.

P1 jeton kazanır (Şekil 4.21). $X_0=[1\ 0\ 0\ 0\ 0\ 0]$



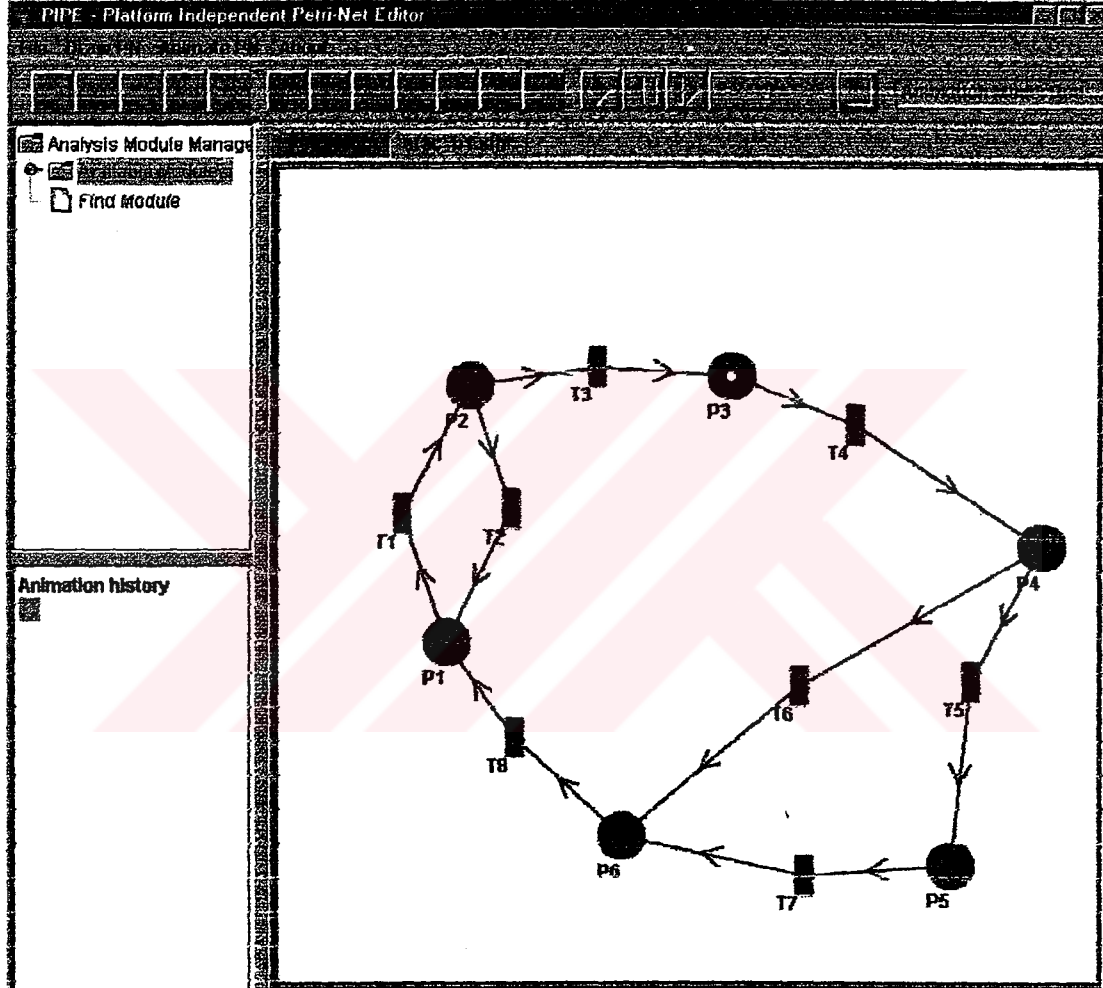
Şekil 4.21 B aracının 4.denemesi için petri ağının 1. aşaması

"Random Fire" seçeneği seçilerek aktif durumdaki T1'in ateşlenmesi sağlanır. P2 jeton kazanır. T3 ve T2 geçişlerinden birisinin seçme aşamasına gelinmiştir (Şekil 4.22). $X_1=[0\ 1\ 0\ 0\ 0\ 0]$



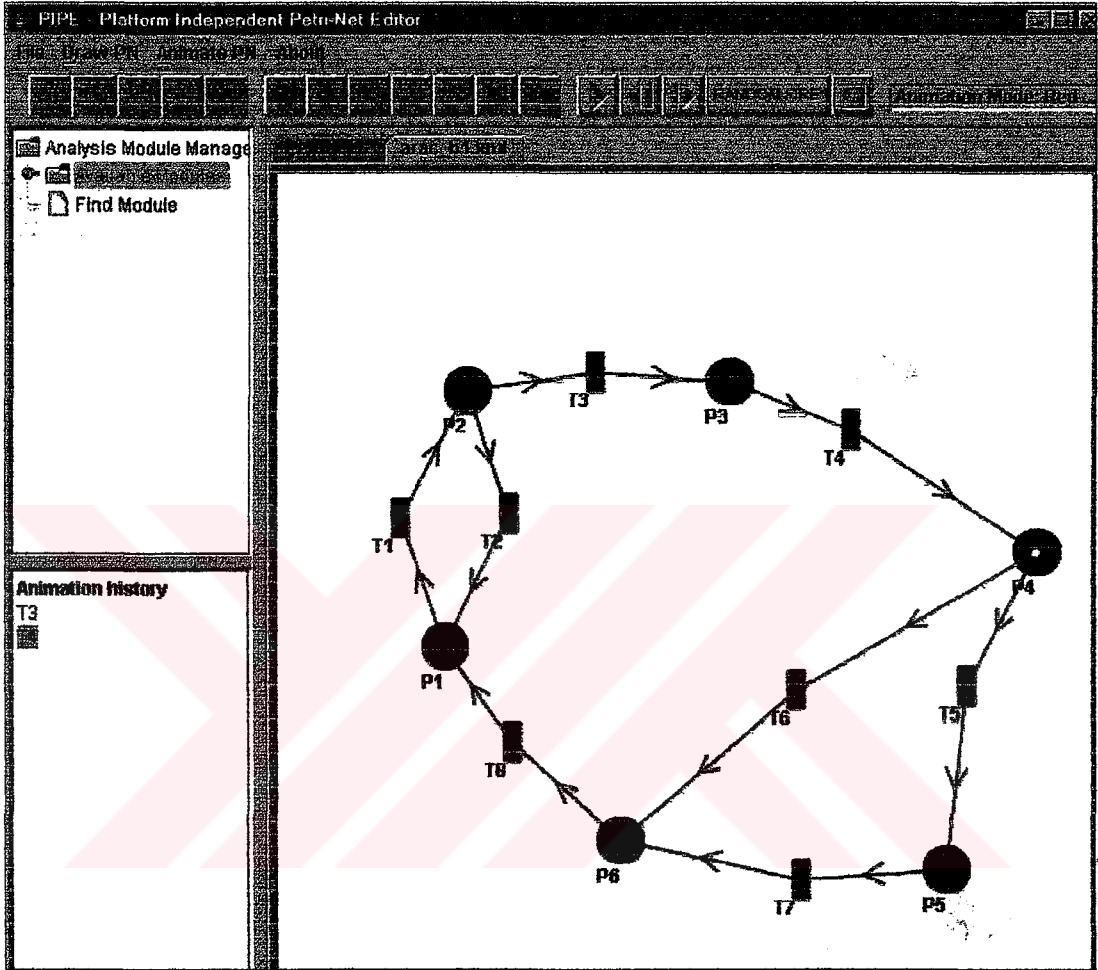
Şekil 4.22 B aracının 4.denemesi için petri ağının 2. aşaması

B aracının Tablo 4.7'deki 4. denemesine göre PIPE'da mousla üzerine tıkl atılarak T3 geçişi seçilir ve T3'ün ateşlenmesi sağlanır. P3 jeton kazanır (Şekil 4.23).
 $X_2=[001000]$



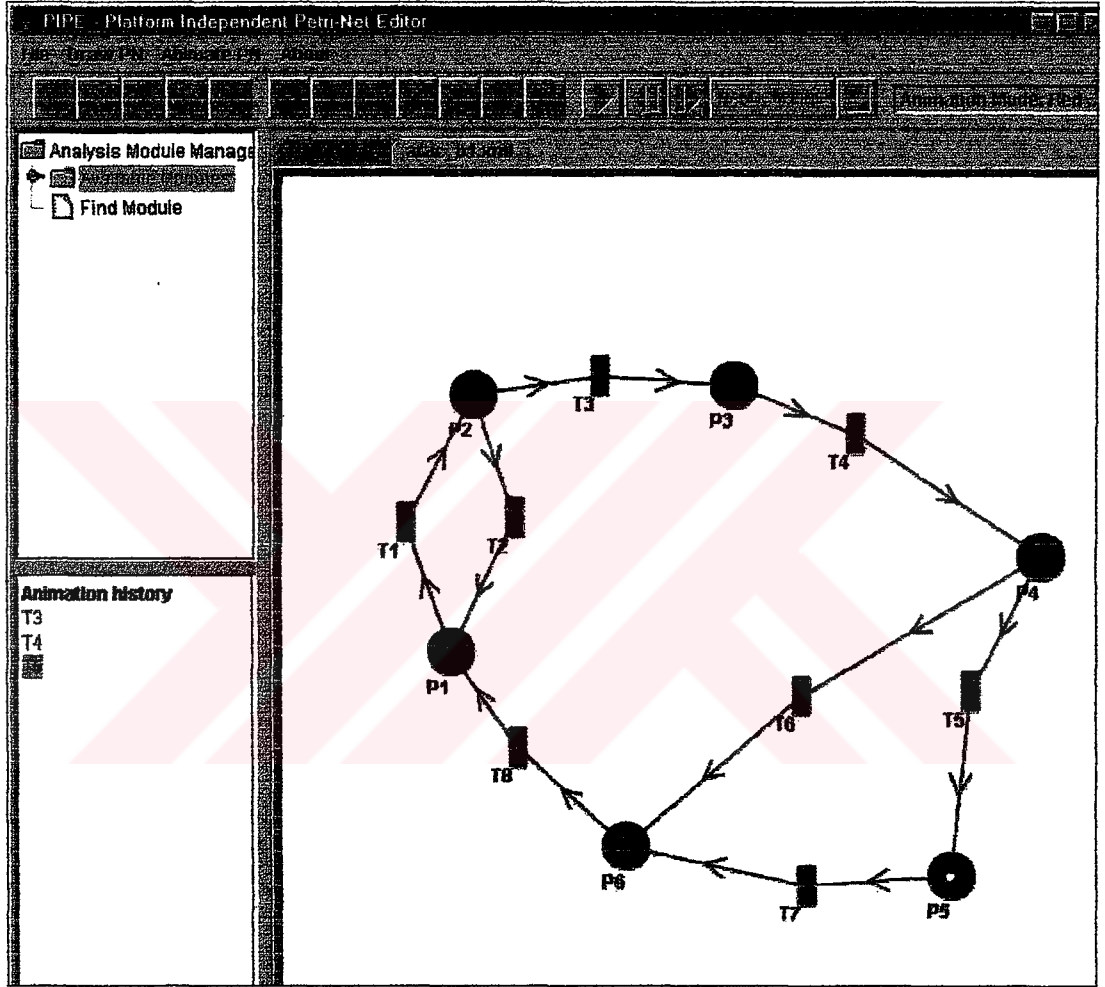
Şekil 4.23 B aracının 4.denemesi için petri ağının 3. aşaması

T4 ateşlenir. P4 jeton kazanır. T6 ve T5 aktif haldedir. Çakışan geçişlerdir. İkisinden birinin seçimi gerekmektedir (Şekil 4.24). $X_3=[0\ 0\ 0\ 1\ 0\ 0]$



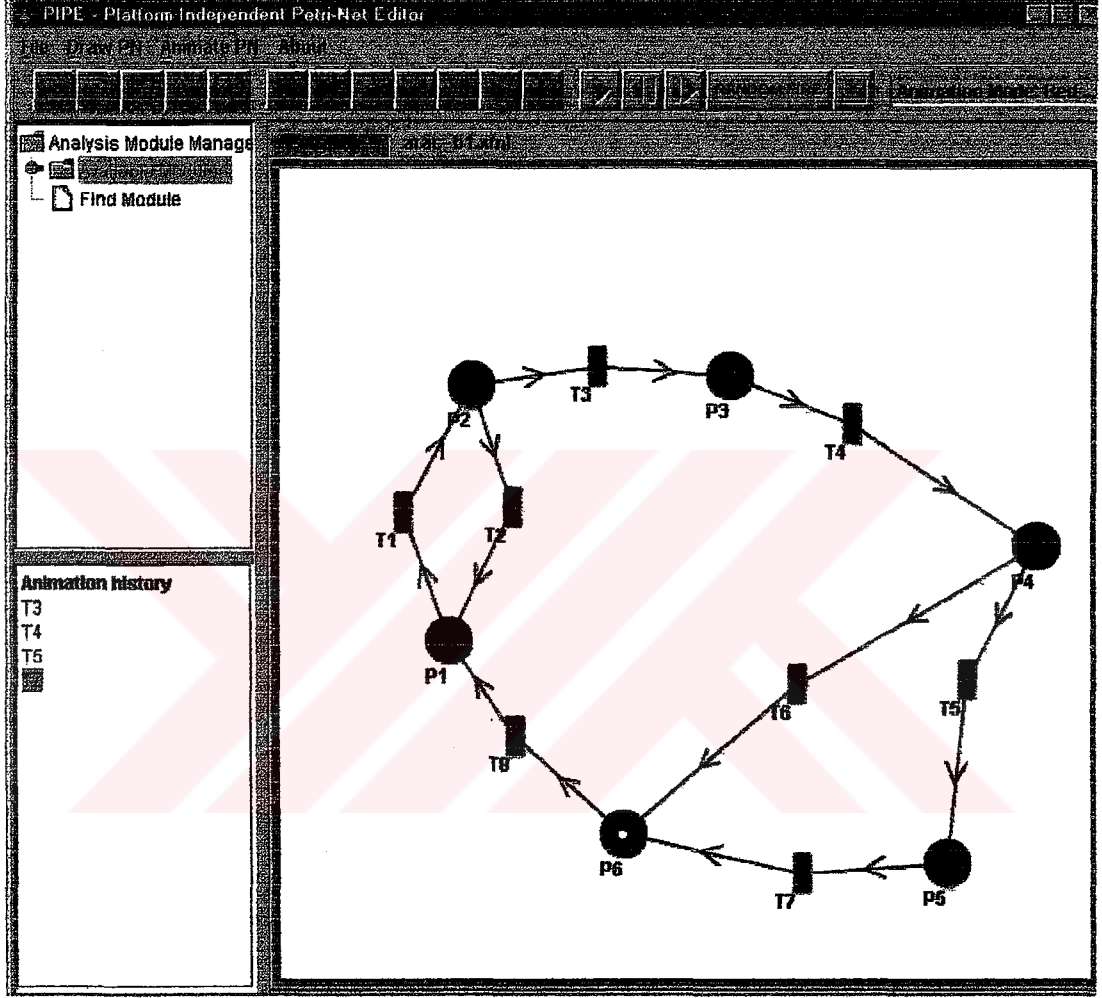
Şekil 4.24 B aracının 4.denemesi için petri ağının 4. aşaması

B aracının Tablo 4.7'deki 4. denemesine göre T5'in üzerine tıklanır ve T5 ateşlenir. P5 jeton kazanır (Şekil 4.25). $X_4 = [0\ 0\ 0\ 0\ 1\ 0]$



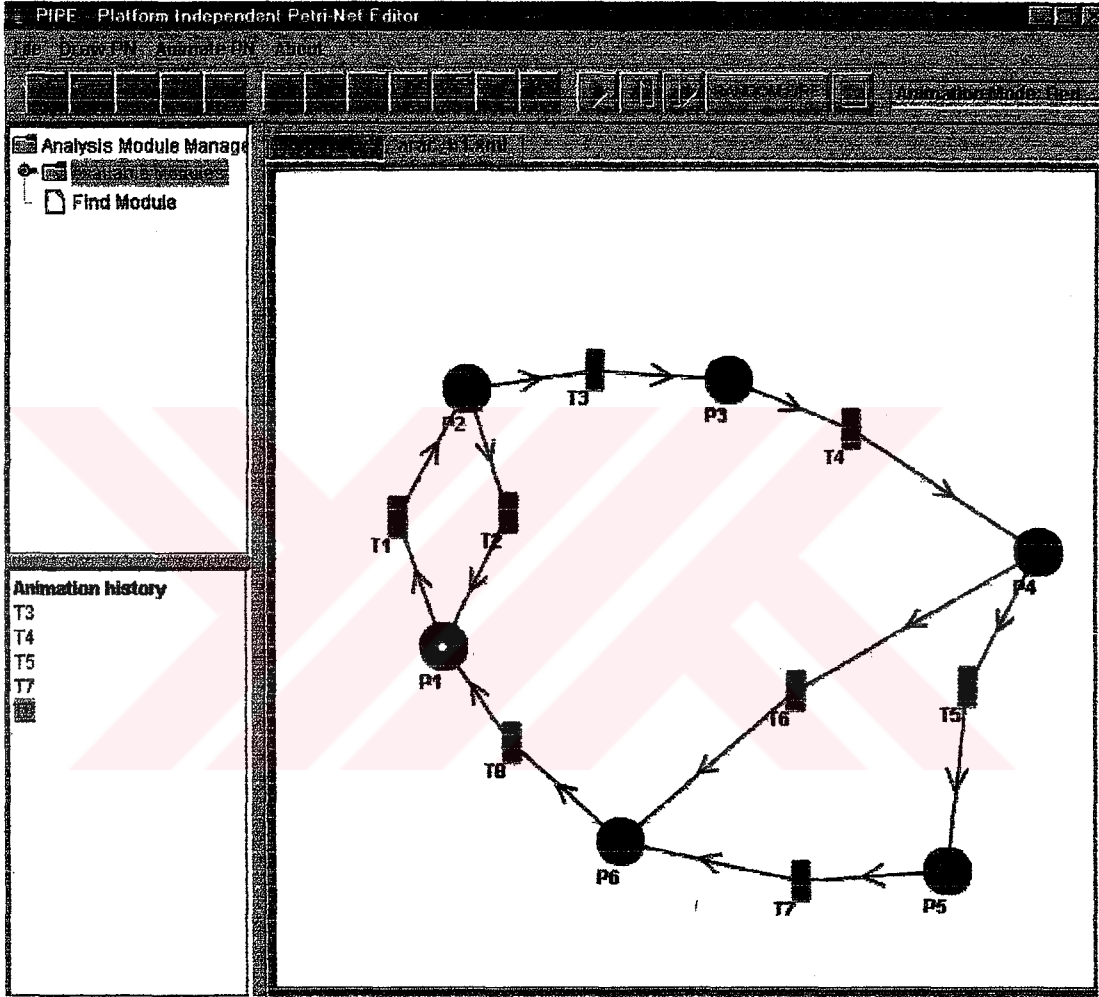
Şekil 4.25 B aracının 4.denemesi için petri ağıının 5. aşaması

T7 üzerine tıklanarak (ya da random olarak) ateşlenir. P6 jeton kazanır (Şekil 4.26).
 $X_5 = [0\ 0\ 0\ 0\ 0\ 1]$



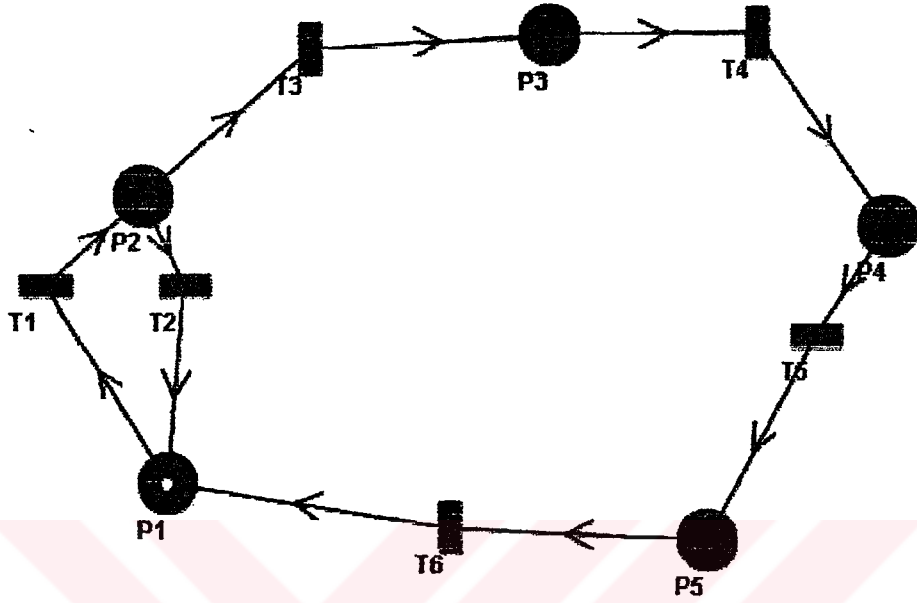
Şekil 4.26 B aracının 4.denemesi için petri ağıının 6. aşaması

T8 tıklanarak (ya da random olarak) ateşlenir ve P1 jeton kazanır. İlk duruma geri dönmüş olur (Şekil 4.27) . $X_6=[1\ 0\ 0\ 0\ 0\ 0]$



Şekil 4.27 B aracının 4.denemesi için petri ağının son aşaması

B aracı için yapılan benzer bir petri ağı A aracı için de hazırlanır. Bu petri ağı şu şekildedir :



Şekil 4.28 A aracı için petri ağı

Şekil 4.28'deki Petri ağındaki Yerler(durumlar) :

P1 : Boşta Durumu

P2 : Meşgul halin ve büyüklüğün kontrol edildiği durum

P3 : Meşgul bayrağını aktif hale gelme durumu

P4 : Onay için bekleme durumu

P5 : Meşgul bayrağını pasif hale getirme ve güncelleme yapma durumu

Geçişler :

T1 : Aracın büyüklüğünün uygun olmadığı veya mesgul bayrağının pasif olduğu zaman ateşlenen geçiş

T2 Aracın büyüklüğünün uygun olmadığı veya mesgul bayrağının pasif olduğu zaman ateşlenen ve GONDER nbilgi_talep_birlesme komutunun gönderildiği geçiş

T3 : GONDER pbilgi_talep_birlesme komutunun gönderildiği geçiş

T4 : AL onay_birlesme komutunun gönderildiği geçiş.

T5 : Boşta hale geçiş.

Şekil 4.28'de gösterilen A aracı için tasarlanan petri ağında P1 yerinde bir jeton bulunduğunda bu A aracının boşta çalıştığını gösterir. T1 geçişi ateşlendiğinde B aracından Birleşme Talebi alınmış olur ve P2 jeton kazanır. Bu yere ulaşıldığında A aracı B'nin meşgul olup olmadığını ve birleşmiş durumdaki takım büyüklüğünün hedef büyüklüğün altında olup olmadığını kontrol eder. Her iki sorgulama için de cevap hayır ise B'ye negatif bilgilendirme gönderir yani T2 geçişi ateşlenir ve P1 jeton kazanır yani A aracı tekrar boşta çalışmaya geri döner. Eğer her iki cevap da evet ise T3 ateşlenir ve P3 jeton kazanır yani A aracı meşgul bayrağını kaldırarak kendinin meşgul durumda olduğunu göstermiş olur. Sonrasında T4 ateşlenir ve B aracına pozitif bilgilendirme gönderir ve P5 jeton kazanır yani onay için bekleme konumuna ulaşılmıştır. Onay geldikten sonra T6 ateşlenir ve P1 jeton kazanır yani A aracı birleşmeyi gerçekleştirmiş ve boşta çalışmaya geri dönmüş olur.

B aracı için yapılan Tablo 4.7'e benzer bir tablo da A aracı için de yapılabilir. A aracı için gerçekleştirilen Petri ağında T2 ve T3 geçişleri çakışmakta ve kullanıcının seçimi gerekmektedir. Bu yüzden burada A'nın sorgulamaları önemlidir. Bunlar: B aracının meşgul olup olmaması ve birleşmiş durumdaki takım büyüklüğünün hedef büyüklüğün altında olup olmadığıdır. Bunlar dikkate alınarak aşağıdaki tabloda gösterildiği gibi A aracı için örnek birleşme durumları oluşturulabilir.

Varsayılanlar : Hedef takım büyüklüğü (N) = 10

Tablo 4.8 A aracına göre B'nin Birleşme Denemeleri Tablosu

B'nin Birleşme Denemesi	B'nin takım büyüklüğü	A'nın takım büyüklüğü	B'nin Durumu	Geçişlerin Ateşlenme Sırası
1.	6	3	Meşgul	T1,T2
2.	8	4	Boşta	T1,T2
3.	9	1	Boşta	T1,T2
4.	2	3	Boşta	T1,T3,T4,T5;T6

Tablo 4.8'den de görüldüğü üzere eğer A'nın takım büyüklüğü + B'nin takım büyüklüğü $< N$ ve B boşta ise T1'den sonra T3 ateşlenir (ve diğer geçişler) eğer takım büyüklükleri toplamı N'e eşit ya da N'den büyük ise veya B aracı mesgul durumda ise T1'den sonra T2 ateşlenir ve A aracı boşta duruma geri döner.

A aracı için PIPE programında gerçekleştirilen ve TEZ CD'sinde bulunan "arac_a.xml" dosyasındaki petri ağında Tablo 4.8'deki örnek denemeler gerçekleştirilebilir. B aracının 4.denemesi için kademe kademe gösterim şeklini ayrıyeten A aracı için yapmayacağız.



5. SONUÇ

Ayrık Olay Sistemlerinde diğer sürekli sistemlerde bilinen zaman kavramı yerine bir olayın gerçekleşme anlarının önemi olduğu için trafik sistemlerinde araçların otoyollara girişi, çıkışı, manevraları hepsi birer olaydır ve ayrık olay sistemi ile modellenenbilirler. Bu yüzden 'Otoyol Giriş Denetimi' ve 'IVHS' konuları incelenmiş ve Ayrık Olay Sistem dillerinden olan Petri Ağlarıyla ilgili dört adet petri ağı uygulaması gerçekleştirilmiştir. Bu konularla ilgili STNP Player yazılım programı ile DES modeli olan Petri ağları ile iki adet uygulama gerçekleştirilmiştir. Ağların tasarımı Simulaworks, Visual Objects Net++, PIPE paket programları yardımıyla gerçekleştirilmiş, içlerinde bulunan "jeton-oyun animasyonları" sayesinde ağların işleyişi gözlemlenmiş, Phyton dilinde yazılmış bir analiz programı yardımıyla ağlardan iki tanesinin Sınırlılık, Canlılık, Sakınım özellikleri analizi yapılmış, Ölü Geçiş, Ulaşılabilirlik kontrolü gerçekleştirilmiştir.

Gelecek yüzyılda otomatik sistemler trafikte sıklıkla kullanılacaktır. Hatta günümüzdeki araba modelleri artık içlerinde bulunan mesafe algılayıcılarıyla sürücünün ufak tefek kazaları engellemesini, daha kolay park etmesini sağlamaktadır. Bu, gelecek yıllarda trafikte, otoyollarda ve araçlarda tamamen otomatikleştirilmiş sistemlere geçileceğini göstermektedir. Ayrık Olay Sistem(DES) yöntemleri ve uygulamaları, geleceğin bu otomatikleştirilmiş yolları ve araçları için etkin çözümler sunmaktadır. Sonuç olarak diyebiliriz ki, gelecek yüzyılda 'otomatikleştirilmiş otoyol ve araçlar' alanında söz sahibi olunabilmesi için Ayrık Olay Sistem uygulamalarına da önem verilmesi gerekmektedir.

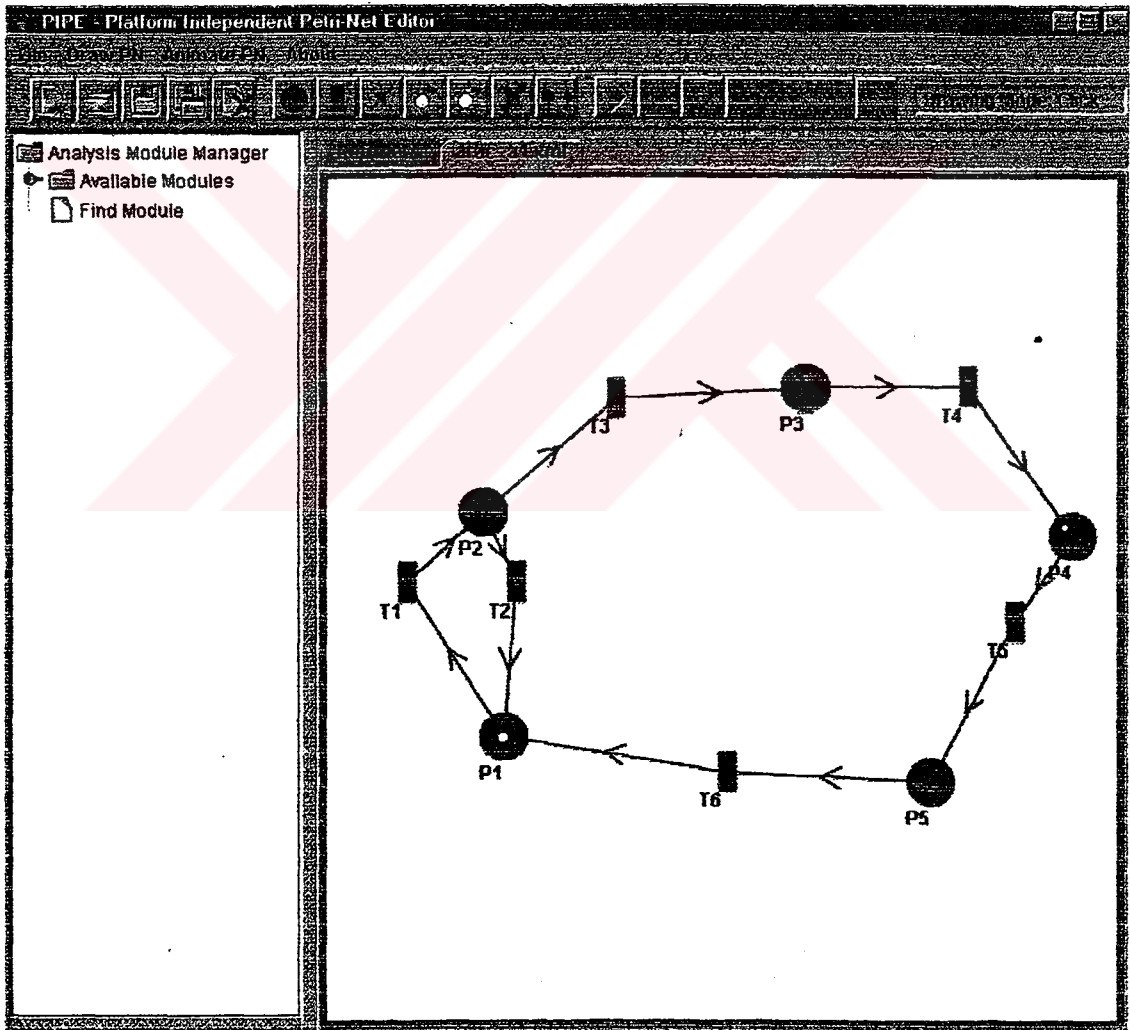
KAYNAKLAR

- [1] **Cassandras G.C., Lafortune S.**, 1999. Introduction To Discrete Event Systems, Kluwer Academic Publishers, Massachusetts.
- [2] <http://www.cs.hmc.edu/courses/2001/spring/cs156/html11/slides11.pdf>
- [3] **Duggan J., Browne J.**, 1988. ESPNET: expert-system-based simulator of Petri nets, *IEEE Proceedings*, Vol.135, No:4, 239-246.
- [4] **Balta M.C.**, 2002. Otoyol Giriş Denetimi, *Yüksek Lisans Tezi*, İ.T.Ü. Fen Bilimleri Enstitüsü, İstanbul.
- [5] <http://science.ankara.edu.tr/~ozbek/bulanik-1.htm>
- [6] <http://www.cs.mcgill.ca/~jhuo/project/>
- [7] **Varaiya P.**, 1993. Smart Cars on Smart Roads : Problems of Control, *IEEE Transactions on Automatic Control*, Vol. 38, No:2, 195-206.
- [8] **Rosen S.**, 1976. Future Facts, Simon and Schuster, New York, 197-199.
- [9] **Gündüz S.**, 1990. Petri ağlarının incelenmesi ve bilgisayar ile simülasyonu, *Yüksek Lisans Tezi*, İ.T.Ü. Fen Bilimleri Enstitüsü, İstanbul.
- [10] <http://www.cise.ufl.edu/~fishwick/introsim/node3.html>
- [11] <http://vita.bu.edu.tr/cgc/MIDEDS/>
- [12] <http://www.simulaworks.com/>
- [13] <http://petri-net.sourceforge.net/>
- [14] http://www.systemtechnik.tu-ilmenau.de/~drath/visual_E.htm
- [15] <http://www.daimi.au.dk/PetriNets/tools/quick.html>

EK A

PLATFORM INDEPENDENT PETRI NET EDITOR (PIPE)

HAKKINDA ÖZET BİLGİ



Şekil A.1 PIPE Ekran Görüntüsü

Web sitesi : <http://petri-net.sourceforge.net/>

Kullanılabilirliği : Ücretsiz

1.1 Programın Özellikleri :

Desteklenen Petri Ağları

- Yer/Geçiş Ağları

Bileşenleri

- Grafik Editör
- Jeton Oyunu Animasyonu
- Hızlı Simülasyon
- Durum Uzayları
- Yoğunlaştırılmış Durum Uzayları
- Yer Sabitleri
- Geçiş Sabitleri
- Yapısal Analiz
- Basit Performans Analizi
- Değiştirilebilir Dosya Formatı
- Genişletilebilir Analiz Modülleri ve Dosya Formatları

1.2 Çevreler

- Java

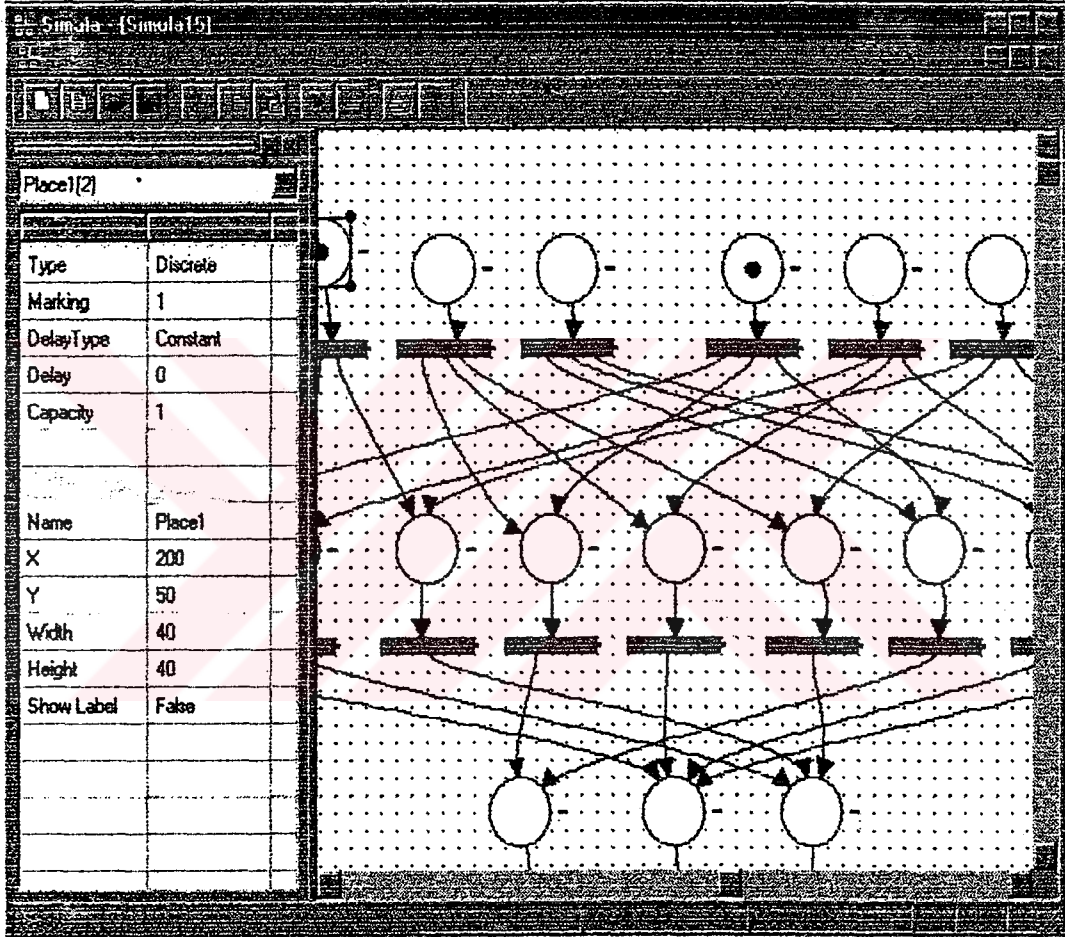
1.3 Program Tanımı

Petri ağlarını hızlı ve etkili şekilde oluşturur..Sabitler Analizi, Durum-Uzayı Analizi ve Simulasyon Analizini de içeren 6 adet analiz modeline sahiptir. Bu tez çalışmasında kullanılmasının asıl sebebi kullanıcıya simulasyon esnasında hangi geçişin ateşleneceğine karar vermesine olanak sağlamasıdır.

EK B

SIMULAWORKS

HAKKINDA ÖZET BİLGİ



Şekil B.1 Simulaworks Ekran Görüntüsü

Web sitesi : <http://www.simulaworks.com/>

Kullanılabilirliği : Ücretsiz

1.4 Programın Özellikleri:

Desteklenen Petri Ağları

- Yer/Geçiş Ağları
- Stokastik Petri Ağları
- Zamanlı Petri Ağları

Bileşenleri

- Grafik Editör
- Jeton Oyunu Animasyonu
- Hızlı Simülasyon

1.5 Çevreler

- PC, MS Windows 95
- PC, MS Windows 98
- PC, MS Windows NT
- PC, MS Windows 2000
- PC, MS Windows XP

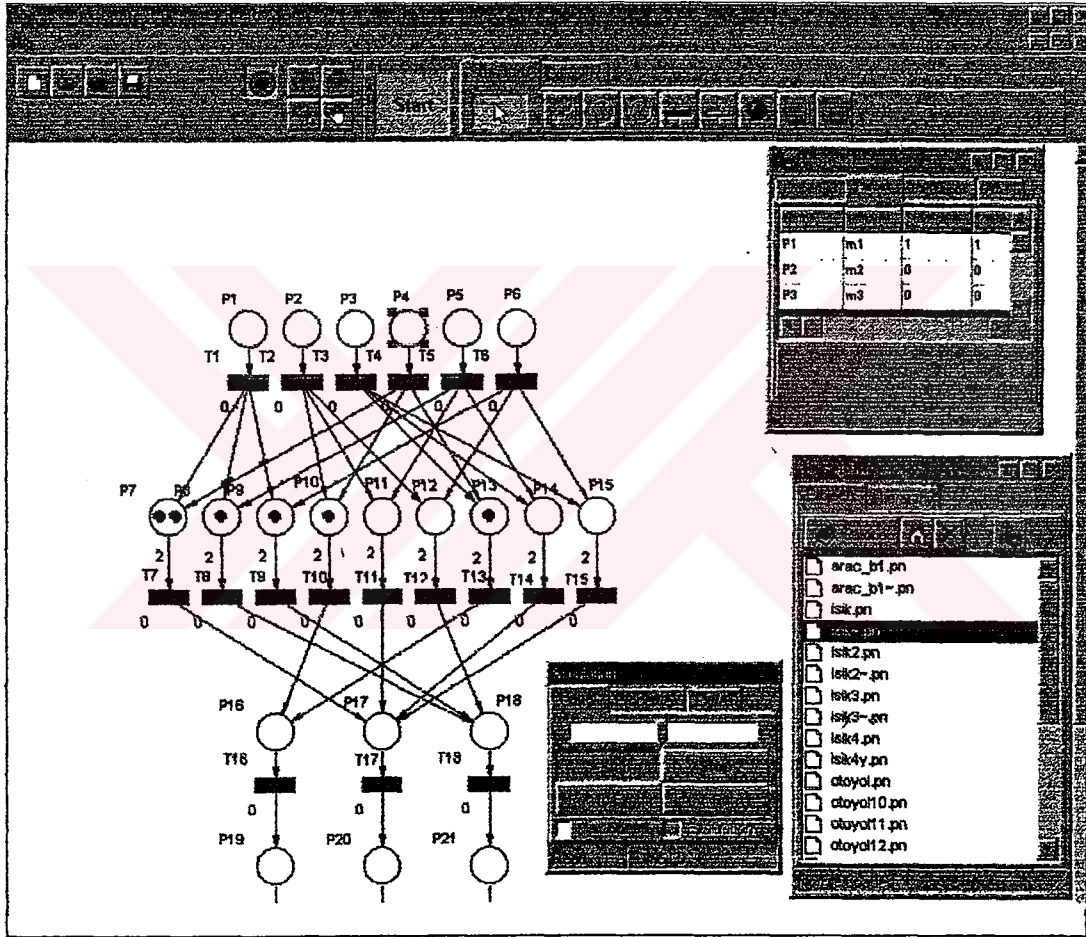
1.6 Program Tanımı

SimulaWorks paketi çoğu simulasyon dillerine ve AI tekniklerine açık genel amaçlı bir paket programdır. Kontrol çevrimleri, lojik ve donanım devreleri, üretim sistemleri, trafik sistemleri vb.'de kullanılabilir. Bu tezde kullanılmasının amacı zamanlı petri ağının modellenmesine olanak tanımasıdır.

EK C

VISUAL OBJECT NET++

HAKKINDA ÖZET BİLGİ



Şekil C.1 Visual Objects Net++ Ekran Görüntüsü

Web sitesi : http://www.systemtechnik.tu-illmenau.de/~drath/visual_E.htm

Kullanılabilirliği : Ücretsiz

1.7 Programın Özellikleri :

Desteklenen Petri Ağları

- Yer/Geçiş Ağları
- Zamanlı Petri Ağları
- Hibrid Dinamik Ağlar ve Hibrit Nesne Ağları

Bileşenleri

- Grafik Editör
- Jeton Oyunu Animasyonu
- Hızlı Simülasyon
- Yapısal Analiz
- Basit Performans Analizi
- Nesne Hiyerarşilerini Destekler.

1.8 Çevreler

- MS Windows

1.9

1.10 Program Tanımı

Visual Object Net ++, PC'ler için sürekli ve ayrık zamanlı olayları destekleyen yenilikçi bir Petri-Ağı CAE programıdır. Bu tezde kullanılmasının amacı petri ağın modelinin işleyişi jeton-oyun animasyonunu güzel bir şekilde canlandırabilmesidir.

EK D

TEZ CD'SİNDE BULUNANLAR

Paket Programlar ve Bunlarda Kullanılan İlgili Dosyalar

- Python 2.3 (python dilindeki dosyalar için gerekli)
 - otoyol.py
 - isik.py
 - textanal.py
 - pnetanal.py
- Java 2 Platform STANDARD EDITION (PIPE için gerekli)
- PIPE
 - arac_a.xml
 - arac_b.xml
- Visual Objects Net++
 - otoyol.pn
- SimulaWorks
 - isik.sim

ÖZGEÇMİŞ

Murat GÜZEL, 1978 yılında Samsun ilinin Bafra ilçesinde doğmuştur. İlk ve orta eğitimini Tekirdağ'ın Çorlu ilçesinde tamamladıktan sonra, 1996 yılında Kırklareli Anadolu Lisesi'nden mezun olmuştur. Aynı yıl İstanbul Teknik Üniversitesi Elektrik Mühendisliği bölümüne girmiştir. Bu bölümden 2000 yılında mezun olmuş ve aynı yıl İstanbul Teknik Üniversitesi Fen Bilimleri Enstitüsü Kontrol-Otomasyon Yüksek Lisans programına başlamıştır.

