ESKİŞEHİR TEKNİK ÜNİVERSİTESİ BİLİM VE TEKNOLOJİ DERGİSİ
B- TEORİK BİLİMLER

Eskişehir Technical University Journal of Science and Technology B- Theoritical Sciences

2018, 6(2), pp. 129 - 139, DOI: 10.20290/aubtdb.421865

BUG-0, 1, 2 ALGORİTMALARI VE PETRİ AĞI MODELLERİ

Alpaslan YUFKA [1], Hanife APAYDIN ÖZKAN [1], Aydın AYBAR [1, *]

[1] Elektrik-Elektronik Mühendisliği, Mühendislik Fakültesi, Anadolu Üniversitesi, 26555, Eskişehir, Türkiye

ÖZET

Bu çalışmada öncelikle, 2B bir ortam içerisinde bilinmeyen engellerin bulunduğu, gidilecek hedefin koordinatlarının bilindiği ve bu hedefin bir başlangıç noktasından ulaşılabilir olduğu bir konfigürasyon içerisinde güzergâh planlaması yapılmasına yönelik Lumelski ve Stepanov tarafından önerilen Bug-0, Bug-1 ve Bug-2 algoritmaları tanıtılmıştır. Sözü edilen algoritmaların işleyişinin gözlemlenmesi uygulanacakları sistemler üzerindeki etkilerinin anlaşılması ve yorumlanması açısından önemlidir. Bu çalışmada söz konusu algoritmaların modellenmesinde kesikli olay sistemlerinin modellenmesi ve analizinde sıklıkla kullanılan Petri Ağları ele alınmış ve algoritmaların işleyişi bu modeller üzerinden ele alınarak işleyişin izlenmesi ve öngörülmesinde büyük kolaylık sağlanmıştır.

Anahtar Kelimeler: Bug-0-1-2 Algoritmaları, Modelleme, Petri ağı

BUG-0, 1, 2 ALGORITHMS AND PETRI NETS' MODELS

ABSTRACT

In this work, Bug-0, Bug-1 and Bug-2 algorithms proposed by Lumelski and Stepanov for route planning in a 2B environment in a configuration with unknown obstacles, where the coordinates of the reachable target from the starting point are known, are introduced. Observation of the operation of these algorithms is important for understanding and interpreting their effects on the applied systems. In this study, Petri nets which are frequently used in modeling and analysis of discrete event systems in the modeling of such algorithms are also discussed and the functioning of the algorithms is handled through these models which provides major convenience to monitor and predict the operation.

Keywords: Bug-0-1-2 Algorithms, Modeling, Petri Net

1. GİRİŞ

Otonom gezgin robotlara yönelik en büyük zorluklardan bir tanesi hareket planlamasıdır [1]. Hareket planı içerisinde, otonom gezgin robota bir hedef verilir ve gezgin robotun bu hedefe ulaşabilmesi için üst seviyede bir planlama yapması beklenmektedir. Üst seviye planlama sonrasında gezgin robotun hareketine yönelik eyleyicilerine alt seviye bir hareket planı ve kontrolü oluşturulmaktadır [1]. Üst seviye planlama için önerilen en eski planlayıcılar BUG algoritmalarıdır [1-5]. BUG algoritmalarının birçok çeşidi olup bu çalışmada Lumelski ve Stepanov tarafından önerilen Bug-0, Bug-1, Bug-2 algoritmaları [1-4] tanıtılmıştır. Bu algoritmalar, 2 Boyutlu (2B) bir ortam içerisinde bilinmeyen engellerin bulunduğu, gidilecek hedefin koordinatlarının bilindiği ve bu hedefin bir başlangıç noktasından ulaşılabilir olduğu bir konfigürasyon içerisinde güzergâh planlaması için kullanılmaktadırlar. Sözü edilen algoritmaların işleyişinin gözlemlenmesi uygulanacakları sistemler üzerindeki etkilerinin anlaşılması ve yorumlanması açısından önemlidir. Petri Ağları kesikli olay sistemlerinin modellenmesinde ve analizinde kullanılan matematiksel ve grafiksel bir modelleme aracıdır [6-8]. Bu çalışmada ele alınan BUG algoritmalarının modellenmesi için de Petri Ağları kullanılmış ve algoritmaların işleyişi bu modeller üzerinden ele alınmıştır. Petri Ağları grafiksel ve matematiksel yönüyle algoritmaların işleyişinin izlenmesinde ve öngörülmesinde büyük kolaylık

* Sorumlu Yazar: aaybar@anadolu.edu.tr
Geliş: 08.05.2018 Kabul: 14.06.2018

sağlamaktadır. Bu çalışmada Bug-1 algoritmasının daha önce önerilen Petri Ağı modelinin [8] geliştirilmiş hali sunulmuş ve Bug-1'e ilaveten Bug-2 algoritmasının Petri Ağı modellemesi verilmiştir.

Bu makalede BUG ailesine ait Bug-0, Bug-1, Bug-2 algoritmaları Bölüm 2'de verilmiştir. Bölüm 3'te ise algoritmaların modellenmesinde kullanılacak Petri Ağları tanıtılmaktadır. Bölüm 4'te Bug-1 ve Bug-2 algoritmalarının Petri Ağı modelleri tanıtılarak, algoritmaların gezgin robot üzerinden işleyişi bu modeller üzerinden ele alınmıştır.

2. BUG ALGORİTMALARI

Klasik güzergâh planlama problemi, Piyano Taşıma Problemi (*Piano Mover's Problem*) ile ele alınmıştır. Bu problemde, bilinen bir ortam içerisinde bilinen engeller arasından piyanonun bir başlangıç noktasından istenen hedef noktaya çarpışmasız bir şekilde taşınmasını sağlayacak güzergâhın planlanması istenmiştir [1]. Bu problemde ortam ve engeller bilinmektedir. Ortam içerisinde bilinmeyen engeller olduğu zaman problem farklı bir noktaya taşınmaktadır. Bu noktada yeni problem, önünde, arkasında, iki yanında dokunsal algılayıcılara sahip bir kaplumbağanın (kaplumbağa bir otomat veya gezgin robot olarak düşünülebilir) labirentten kaçması şekline dönüşmüştür. John Pledge bu amaçla Pledge Algoritmasını (*Pledge Algorithm*) geliştirmiştir [4, 9]. Bu algoritmada, başlangıçta keyfi bir yön (kuzey, güney, doğu, batı) seçilmektedir. Sonrasında engelle karşılaşıncaya kadar seçilen yönde ilerlenmektedir. Karşılaşılan engel, otomatın sağında olacak şekilde manevra yapılır ve toplam dönüş sıfır oluncaya kadar (otomatın baş yönünün tekrardan seçilen yönde olması) engel sağ tarafta olacak şekilde engelin sınırları takip edilir. Toplam dönüş sıfır olunca tekrardan seçilen yönde ilerlenmeye devam edilmektedir. Ancak, bu algoritma labirentin içinde veya dışında yer alacak sabit bir noktayı bulmak için kullanılamamaktadır. Otomat, labirentte kolaylıkla sonsuz bir döngüye (tuzak) girebilmektedir [4, 9]. BUG algoritmaları, temelde sözü edilen bu algoritmalar üzerine kurulmuş ve böceklerden esinlenmiştir. Böcekler (kalorifer böceği, hamam böcekleri) sahip oldukları antenlerle çevreyi dokunarak algılayabilmekte ve bir yere ulaşabilmek için önüne çıkan engellerin çevresinden dolanmaktadırlar. Lumelski ve Stepanov tarafından önerilen Bug-0, Bug-1 ve Bug-2 algoritmaları, algılayıcı tabanlı (dokunsal) en eski planlayıcılardır ve basitliklerinden dolayı bu algoritmalar düşük bir hafızaya ihtiyaç duymaktadırlar [1, 4].

Bug-0, Bug-1 ve Bug-2 algoritmaları gezgin bir robotun bilinmeyen 2B bir ortam içerisinde hedefine varması için kullanılırlar [1-5]. Bu algoritmalar ortama ait sadece yerel bilgiyi ve sadece varmak istenen küresel bir hedef noktayı dikkate alırlar. Yerel bilgiyi ve varmak istenen hedef noktayı kullanan gezgin robot bilinmeyen 2B bir ortam içerisinde bir engel ile karşılaştığında, engelin çevresini dolaşacak şekilde bir planlama yapmaktadır. BUG algoritmalarında ana amaç, gezgin robotun engelin sınırlarını takip edecek şekilde çarpışmasız bir güzergâh oluşturmasını ve gezgin robotun küresel hedef noktaya doğru yönelim göstermesini sağlamasıdır. Bunlara ilave olarak, BUG algoritmaları geleneksel olarak üç temel varsayıma sahiptir: ***i)*** gezgin robot bir nokta olarak kabul edilir, ***ii)*** gezgin robot mükemmel bir konumlamaya sahiptir ve ***iii)*** gezgin robotun algılayıcıları kusursuzdur [1-5].

İzleyen alt başlıklarda BUG ailesine ait Bug-0, Bug-1, Bug-2 algoritmaları [1-5] anlatılacaktır.

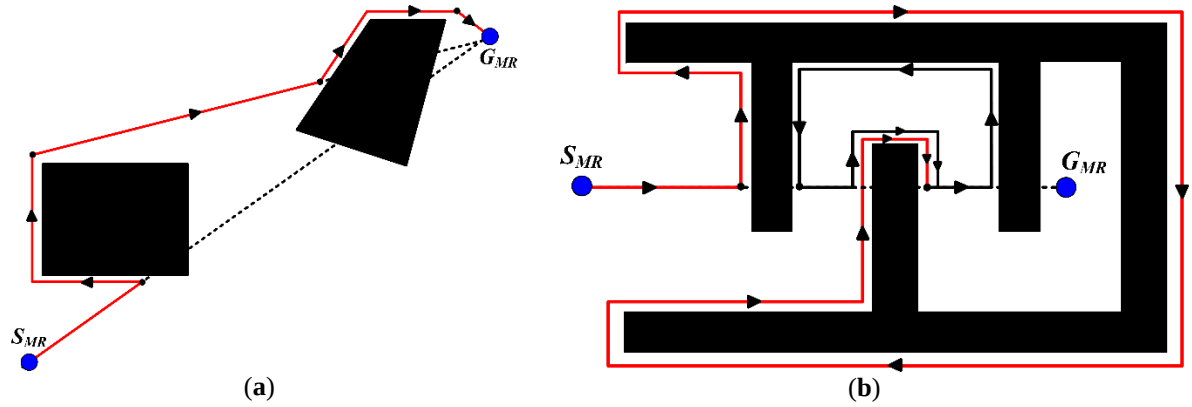
2.1. Bug-0 Algoritması

Bug-0 algoritması en ilkel planlayıcı olup herhangi bir hafıza ihtiyacı bulunmamaktadır. Gezgin robot (MR), başlangıç noktası $S_{MR} = (x_S, y_S)$'yi ve varılmak istenen hedef noktayı $G_{MR} = (x_G, y_G)$'yi bilmektedir. Noktalara ait koordinatlar 2B Kartezyen Koordinat Sistemi'nde verilmektedir. Bug-0 algoritmasında [4]:

Adım 1: Aşağıdaki koşullar sağlanmadıkça, direkt olarak hedef noktası G_{MR}'ye doğru ilerle.
Adım 1.1: Hedefe ulaşıldı. Prosedürü durdur.
Adım 1.2: Engel ile karşılaşıldı. *Adım 2*'ye geç.

Adım 2: Aşağıdaki koşullar sağlanmadıkça, kabul edilen yönde (saat yönünde/yönünün tersine (varsayılan)) engelin sınırlarını takip et (duvar takibi davranışı).
Adım 2.1: Hedefe ulaşıldı. Prosedürü durdur.
Adım 2.2: Gezgin robotun baş açısı hedefe doğru bakıyor ve önü açık. *Adım 1*'e geç.

Bug-0 algoritmasına yönelik gezgin robotun davranışı Şekil 1'de örneklenmiştir. Şekil 1 içerisindeki kırmızı yol gezgin robotun engelin sınırlarını takip ettiği yolu göstermektedir. Siyah renkli yol ise Bug-0 algoritmasının yerel bir çevrimde sonsuz döngüye girdiği yolu göstermektedir. Yerel çevrim, iç bükey bir nesnenin içerisinde gezgin robotu bir döngüye sokacak bir yapı olarak düşünülebilir. Şekil 1.(b)'de görüldüğü üzere Bug-0 algoritması, Pledge Algoritmasında [7] olduğu gibi yerel bir çevrimden dolayı kolayca sonsuz bir döngüye girebilmektedir.



Şekil 1. İki farklı 2B ortam için Bug-0 algoritmasının üreteceği güzergâhlar: (a) Birinci ortam (2 bilinmeyen engel, yerel çevrimsiz); (b) İkinci ortam (tek engel, yerel çevrimli).

2.2. Bug-1 Algoritması

Bug-1 algoritması, Bug-0'a göre daha gelişmiş bir algoritmadır ve 3 adet kayıtçıya sahip bir planlayıcıdır. Bu kayıtçılar, R_1, R_2 ve R_3 olarak anılmaktadır. Gezgin robot'un anlık konumu 2B Kartezyen Koordinat Sistemi'nde ifade edilmektedir ve $Q_{MR} = (x_{MR}, y_{MR})$ olarak gösterilmektedir. Gezgin robot hedefe ilerlerken engel ile karşılaşması durumunda, karşılaşılan nokta, çarpışma noktası $H_j := (x_{H_j}, y_{H_j})$ ve bu engeli terk edeceği nokta, ayrılış noktası $L_j := (x_{L_j}, y_{L_j})$ olarak anılmaktadır. L_0 noktası gezgin robotun başlangıç noktasıdır $S_{MR} = (x_S, y_S)$. R_1 kayıtçısı, Q_{MR} ile G_{MR} arasındaki en küçük mesafeyi verecek Q_{MR} noktasının koordinatlarını tutmaktadır. R_2 kayıtçısı, H_j noktasından itibaren gezgin robotun engelin sınır uzunluğu boyunca kat ettiği yolu tutmaktadır. R_3 kayıtçısı, Q_{MR} noktasından (R_1 kayıtçısı içerisinde tutulan Q_{MR} noktası alınabilir) itibaren gezgin robotun engelin sınır uzunluğu boyunca kat ettiği mesafeyi tutmaktadır. Bug-1 algoritmasında [1, 4]:

Adım 1: Aşağıdaki koşullar sağlanmadıkça, L_{j-1} noktasından (başlangıçta S_{MR} ve $j = 1$) G_{MR} noktası boyunca düz bir güzergâh boyunca direkt olarak G_{MR}'ye doğru ilerle.
Adım 1.1: G_{MR}'ye ulaşıldı. Prosedürü durdur.
Adım 1.2: Engel ile karşılaşıldı. Bir çarpışma noktası H_j belirle. *Adım 2*'ye geç.

Adım 2: Aşağıdaki koşullar sağlanmadıkça, kabul edilen yönde (saat yönünde/yönünün tersine (varsayılan)) engelin sınırlarını takip et (duvar takibi davranışı).

Adım 2.1: G_{MR}'ye ulaşıldı. Prosedürü durdur.

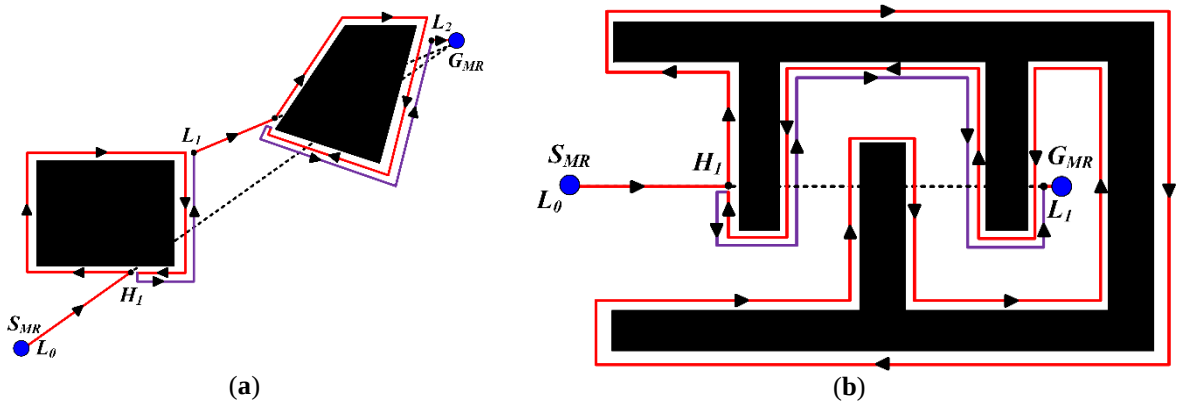
Adım 2.2: R_1, R_2 ve R_3 kayıtçılarını çalıştır. Engelin tüm sınırlarını dolaşıp tekrardan H_j noktasına gelindiğinde R_1 içerisinde tutulan Q_{MR} noktasını, yeni ayrılış noktası L_j olarak ata. *Adım 3*'e geç.

Adım 3: R_2 ve R_3 içerisinde tutulan mesafe bilgilerini karşılaştır.

Adım 3.1: H_j noktasından L_j noktasına en kısa mesafede gidebilmek için engeli takip edeceği yönü (saat yönünde/yönünün tersine) belirle.

Adım 3.2: L_j noktasına ulaşıldı. j indeksini $j = j + 1$ olarak artır. Engelden ayrıl. *Adım 1*'e geç.

Bug-1 algoritması sonucu oluşacak güzergâhın uzunluğu en fazla $d(S_{MR}, G_{MR}) + \frac{3}{2}\sum_{i=1}^{N} p_i$ kadar ve en az $d(S_{MR}, G_{MR})$ kadar olabilmektedir. Burada; $d(S_{MR}, G_{MR}) := \sqrt{(x_S - x_G)^2 + (y_S - y_G)^2}$, S_{MR} noktasından G_{MR} noktasına olan Öklid uzaklığını göstermektedir ve p_i ifadesi toplamda karşılaşılan N adet engelden i'inci engelin çevre uzunluğunu belirtmektedir. Ortamda herhangi bir engel bulunmuyorsa, üretilen güzergâh, en kısa yol olan Öklid uzaklığı (Kuş bakışı uzaklık) olacaktır. Bug-1 algoritması, hedef ulaşılabilir olduğu sürece 2B ortamda herhangi bir noktadan başlatılabilmektedir. Bug-1 algoritmasına yönelik gezgin robotun davranışı Şekil 2'de örneklenmiştir. Şekil 2 içerisindeki kırmızı yol gezgin robotun engelin sınırlarını ilk kez takip etmeye başladığı yolu göstermektedir. Mor renkli yol ise ayrılma noktasına kadar olan geri dönüş yolunu göstermektedir.



Şekil 2. İki farklı 2B ortam için Bug-1 algoritmasının üreteceği güzergâhlar: (a) Birinci ortam (2 bilinmeyen engel, yerel çevrimsiz); (b) İkinci ortam (tek engel, yerel çevrimli).

2.3. Bug-2 Algoritması

Bug-2 algoritması, Bug-1'e nazaran daha sabırsız bir algoritmadır. Bug-2 algoritmasında en başta S_{MR}'den G_{MR}'ye doğru Öklid uzaklığı boyunca bir güzergâh belirlenir ve gezgin robot G_{MR}'ye varıncaya kadar her fırsatta bu güzergâhı izlemektedir. Bu güzergâh *M*-çizgisi olarak anılmaktadır. *M*-çizgisi'nin eğimi $(x_G - x_S)/(y_G - y_S)$'ye eşittir. Gezgin robot *M*-çizgisi üzerinde olup olmadığını anlık Q_{MR} noktası ile G_{MR} noktası arasındaki eğime bakarak anlayabilmektedir. Gezgin robotun G_{MR} ile olan anlık eğimi $(x_G - x_{MR})/(y_G - y_{MR})$ ile bulunabilmektedir. Bug-2 algoritmasında [1, 4]:

Adım 1: Aşağıdaki koşullar sağlanmadıkça, L_{j-1} noktasından (başlangıçta S_{MR} ve $j = 1$) *M*-çizgisinin belirttiği güzergâh boyunca direkt olarak G_{MR}'ye doğru ilerle.

Adım 1.1: G_{MR}'ye ulaşıldı. Prosedürü durdur.

Adım 1.2: Engel ile karşılaşıldı. Bir çarpışma noktası H_j belirle. *Adım 2*'ye geç.

Adım 2: Aşağıdaki koşullar sağlanmadıkça, kabul edilen yönde (saat yönünde/yönünün tersine (varsayılan)) engelin sınırlarını takip et (duvar takibi davranışı).

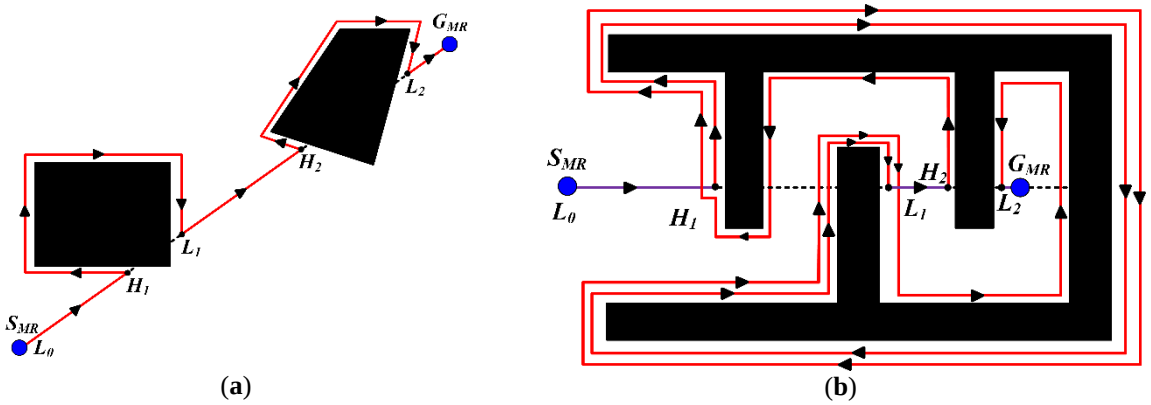
Adım 2.1: G_{MR}'ye ulaşıldı. Prosedürü durdur.

Adım 2.2: Gezgin robot *M*-çizgisi üzerinde ise $d(Q_{MR}, G_{MR})$ Öklid uzaklığını hesapla.

Adım 2.3: $d(Q_{MR}, G_{MR})$ Öklid uzaklığı, $d(H_j, G_{MR})$ Öklid uzaklığından küçük ise *Adım 3*'e geç. Değilse engelin sınırlarını takip etmeye devam et.

Adım 3: Gezgin robotun *M*-çizgisi üzerindeki Q_{MR} noktasını, yeni ayrılış noktası L_j olarak ata. j indeksini $j = j + 1$ olarak artır. Engelden ayrıl. *Adım 1*'e geç.

Bug-2 algoritması sonucu oluşacak güzergâhın uzunluğu en fazla $d(S_{MR}, G_{MR}) + \frac{1}{2}\sum_{i=1}^{N}(n_i.p_i)$ kadar ve en az $d(S_{MR}, G_{MR})$ kadar olabilmektedir. Burada, N adet engelden i'inci engel *M*-çizgisi tarafından n_i sayıda kesişebilmektedir. n_i sayıda kesişimin her biri i'inci engelden bir ayrılış noktasıdır. Bug-2 algoritması, hedef ulaşılabilir olduğu sürece 2B ortamda herhangi bir noktadan başlatılabilir. Bug-2 algoritmasına yönelik gezgin robotun davranışı Şekil 3'te örneklenmiştir. Şekil 3 içerisindeki kırmızı yol gezgin robotun çarpışma noktasından bir sonraki noktaya ilerlediği güzergâhı göstermektedir. Mor renkli yol ise gezgin robotun ayrılış noktasından bir sonraki noktaya ilerlediği yolu göstermektedir.



Şekil 3. İki farklı 2B ortam için Bug-2 algoritmasının üreteceği güzergâhlar: (a) Birinci ortam (2 bilinmeyen engel, yerel çevrimsiz); (b) İkinci ortam (tek engel, yerel çevrimli).

3. PETRİ AĞLARI

Bir Petri Ağı, daire ile gösterilen yerler, çubuk ile gösterilen geçişler ve yerlerden geçişlere veya geçişlerden yerlere doğru yönlendirilmiş oklardan oluşur. Yer, bir işlemi veya bir kaynağın durumunu; geçiş ise, bir olayın ya da bir işlemin, başlangıcını ve/veya bitişini göstermektedir Bir Petri Ağının yapısında yerler ve geçişlere ek olarak bulunduğu yerlerde işlemin yapılıyor olduğunu ifade eden "●" şeklinde gösterilen belirtiler vardır. Bir yerde belirti olup olmamasına veya sayısına göre ilgili yerlerin gösterdiği kaynak veya işlem hakkında bilgi sahibi olunur. Burada kaynak, sistemde gerçekleştirilmek istenen olaylar için gerekli olan koşulları, kullanılan girdileri makineleri veya personeli ifade etmektedir. Belirtiler mantıksal durumu ifade etmek için de kullanılır. Bir yerde belirti bulunması, şartın sağlandığı bulunmaması ise, şartın sağlanmadığı anlamına da gelir [6-8].

Bir Petri Ağı $G(P, T, N, O, M_0)$ şeklinde gösterilir. Burada $P \coloneqq \{p_1, p_2, \ldots, p_m\}$ yerlerin kümesini şeklinde ve $T \coloneqq \{t_1, t_2, \ldots, t_n\}$ geçişlerin kümesini, $N: PxT \rightarrow \mathbb{N}$ yerlerden geçişlere doğru olan bağlantıların ağırlıklarını gösteren girdi matrisini, $O: PxT \rightarrow \mathbb{N}$ geçişlerden yerlere doğru olan bağlantıların ağırlıklarını gösteren çıktı matrisini ifade etmektedir ($\mathbb{N}$ doğal sayılar kümesidir.). Bir Petri Ağının belirttiği sistemin durumu $M: P \rightarrow \mathbb{N}$ işaretleme vektörü ile ifade edilirken, M_0 başlangıç anındaki işaretleme vektörü ile gösterilmektedir [6-8].

Petri Ağında, bir $t \in T$ geçişine bağlı tüm girdi yerleri $\bullet t = \{p \in P | N(p,t) \neq 0\}$ ile ifade edilir. Bir geçiş için $M(p) \geq N(p,t) \geq 1, \ \forall p \in \bullet t$ koşulunun sağlanması durumunda, bu t geçişi ateşlenebilmektedir. Burada, $M(p)$ ile belirtilen M işaretleme vektöründeki p'inci yerin belirti sayısını vermektedir. M işaretleme vektörü için aktif olan geçişler kümesi E ile gösterilmektedir. Aktif olan bir $t \in E$ geçişi ateşlendiğinde bir sonraki işaretleme vektörü M', Eşitlik (1)'de belirtildiği şekilde hesaplanmaktadır. M_0 işaretleme vektöründen itibaren elde edilen tüm yeni ve farklı vektörler $R(M_0)$ ulaşılabilirlik seti içerisinde tutulur. Set içerisindeki durumların birbirleri arasında oluşturdukları ilişkiler ulaşılabilirlik ağacı olarak gösterilir [6-8].

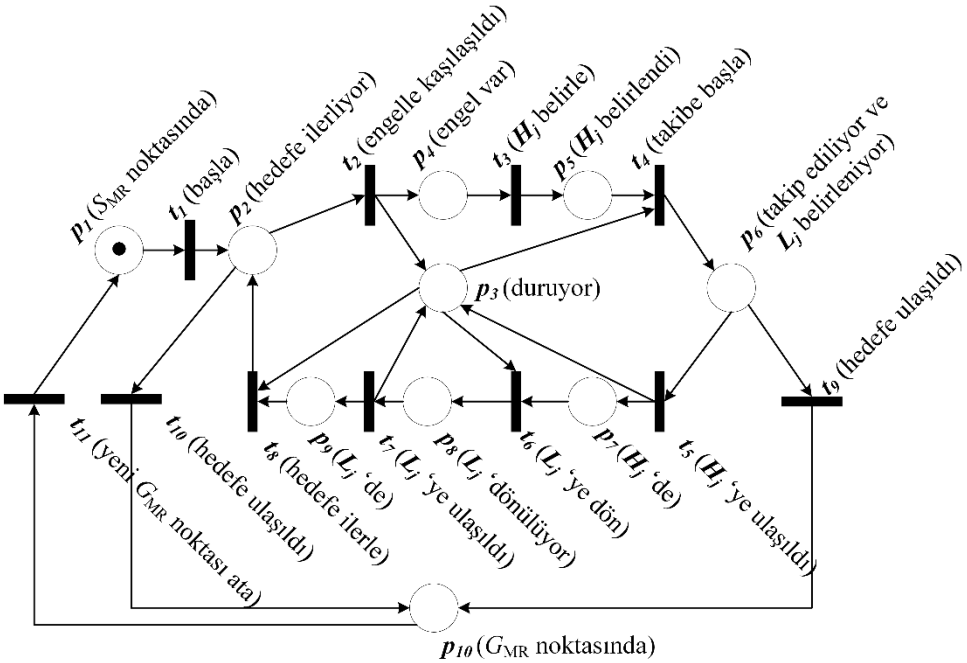
$$M'(p) = M(p) + O(p,t) - N(p,t), \forall p \in P \tag{1}$$

4. BUG-1 VE BUG-2 ALGORİTMALARI İÇİN OLUŞTURULAN PETRİ AĞI MODELLERİ

Bu başlık altında Bug-1 ve Bug-2 algoritmalarının Petri Ağı modellenmesine yönelik çalışmalar sunulacaktır. [8]'de Bug-1 algoritmasına yönelik bir Petri Ağı modeli daha önceden sunulmuştur. Bu çalışma içerisinde sunulacak Bug-1 algoritmasına yönelik Petri Ağı modeli [8]'de sunulan modele nazaran daha üst seviyede bir modeldir. [8] içerisinde sensörlerin engelleri algılayıp algılamadığı ve sensör denetimi gibi alt seviye hususlar bulunmaktadır. Alt seviyeye yönelik arayüz ve kontrollerin üst seviyede yer almaması modelin anlaşılabilirliğini artırmıştır. Bug-1'e ilaveten Bug-2 algoritmasının da Petri Ağı modellemesi verilecektir.

4.1. BUG-1 Algoritması için Petri Ağı Modeli

Bölüm 2.2. içerisinde verilen Bug-1 algoritmasına yönelik hazırlanan Petri Ağı modeli Şekil 4'te verilmiştir.



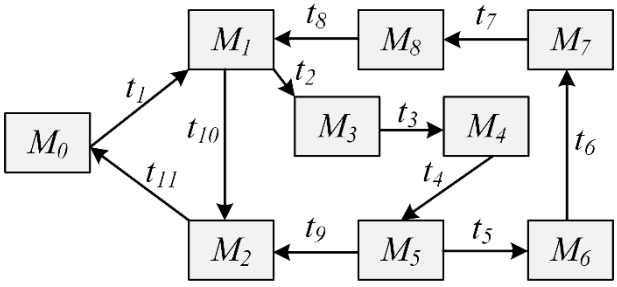
Şekil 4. Bug-1 algoritmasına yönelik hazırlanan Petri Ağı modeli

Bu model içerisinde tanımlanan Petri Ağında yerlerin kümesi $P := \{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9, p_{10}\}$ ve geçişlerin kümesi $T := \{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}, t_{11}\}$'dur. Burada yerlerin fiziksel anlamları şu şekilde tanımlanmaktadır. p_1 yeri, gezgin robotun S_{MR} noktasında olduğunu gösterir. p_2 yeri, gezgin

robotun G_{MR}'ye doğru hareket halinde olduğunu gösterir. p_3 yeri, gezgin robotun hareketsiz olduğunu gösterir. p_4 yeri, bir engel ile karşılaşıldığını gösterir. p_5 yeri, H_j noktasının belirlendiğini ifade eder. p_6 yeri, engelin sınırlarının takip edildiğini ve takip sırasında L_j noktasının belirlenmeye devam edildiğini gösterir. p_7 yeri, H_j noktasına geri dönüldüğünü gösterir. p_8 yeri, daha önceden izlenen en kısa güzergâhtan L_j noktasına geri döndüğünü göstermektedir. p_9 yeri, L_j noktasına varıldığını gösterir. p_{10} yeri, G_{MR} noktasına varıldığını gösterir. T kümesi içerisindeki geçişlerin kümesi de şu şekilde tanımlanmaktadır. t_1 geçişi, Bug algoritmasını başlatmak için kullanılan olaydır. t_2 geçişi, gezgin robotun bilinmeyen bir engel ile karşılaştığını belirten olaydır. t_3 geçişi, H_j noktasının belirlenildiği olaydır. t_4 geçişi, gezgin robotun engelin çevresini takip etmeye başladığı olaydır. t_5 geçişi, H_j noktasına varıldığı olaydır. t_6 geçişi, gezgin robotun L_j noktasına geri dönmeye başladığını gösteren olaydır. t_7 geçişi, L_j noktasına varıldığı olaydır. t_8 geçişi, gezgin robotun L_j noktasından G_{MR}'ye doğru ayrıldığını gösteren olaydır. t_9 ve t_{10} geçişleri, G_{MR} noktasına varıldığı olaylardır. t_{11} geçişi, gezgin robota yeni hedef noktası G_{MR}'nin atamasını yapan olaydır. Geçişlerde sözü edilen konuma yönelik olaylar gezgin robotun gerçek zamanlı konumlama sistemi ile tetiklenmektedir. Ayrıca engel algılamaya yönelik olaylar sonar algılayıcılar tarafından tetiklenmektedir. Gezgin robot başlangıçta S_{MR} noktasındadır. Bu durum $\boldsymbol{M}_0 = [1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T$ işaretleme vektörü ile ifade edilmektedir ($[\ ]^T$ ifade transpozeyi ifade etmektedir.). Bug-1 için oluşturulan Petri Ağının ulaşılabilirlik kümesi 9 adet durum içermektedir. Ulaşılabilirlik kümesi $R = \{\boldsymbol{M}_0, \boldsymbol{M}_1, \dots, \boldsymbol{M}_8\}$ içerisindeki durumların karşılıkları ve fiziksel anlamları Tablo 1'de verilmiştir. Durumlar arası ilişkiyi gösteren ulaşılabilirlik ağacı da Şekil 5'te gösterilmiştir.

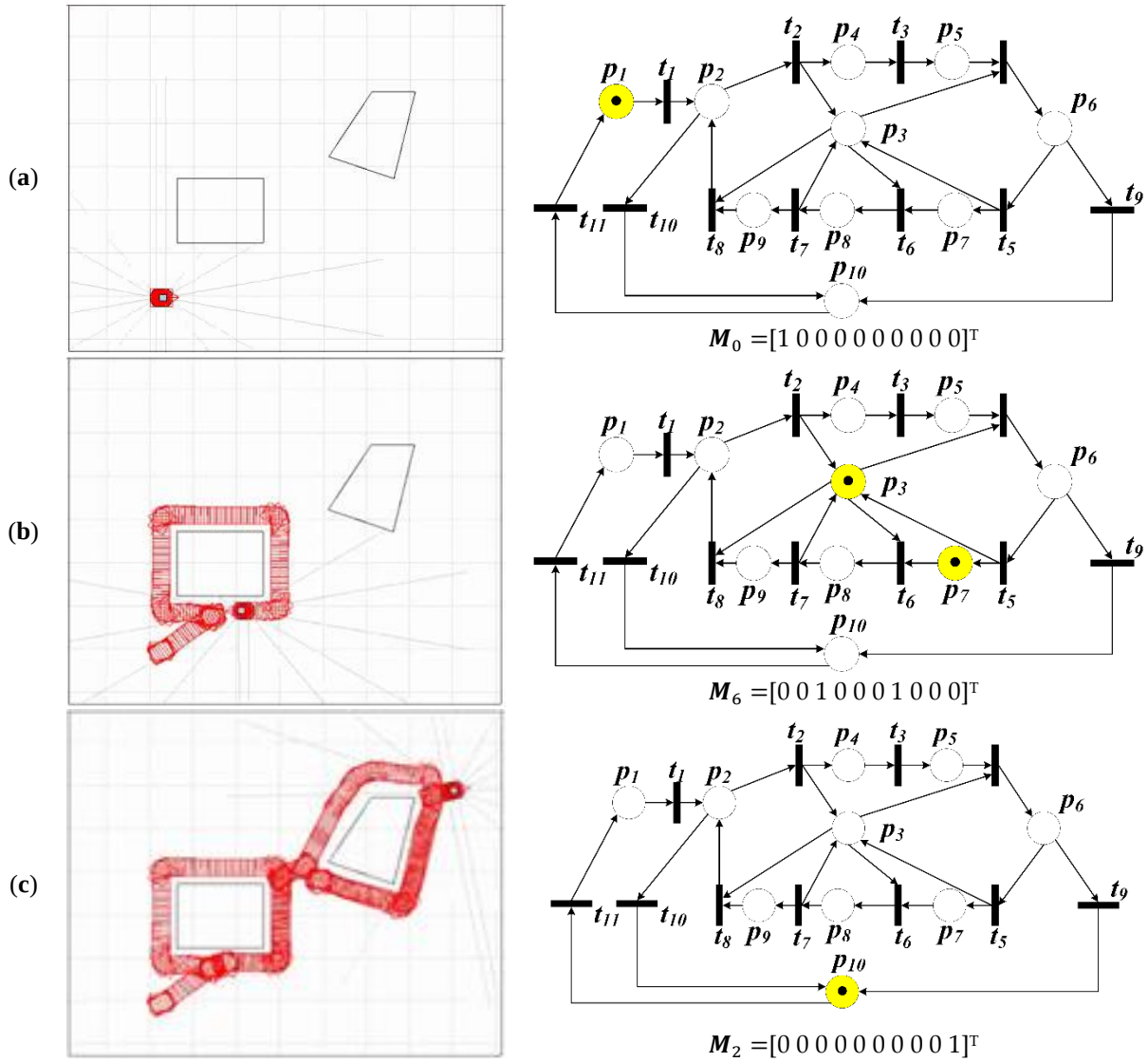
Tablo 1. Bug-1 algoritması Petri Ağı modeli için gezgin robotun tüm durumları

Durum Etiketi	İşaretleme Vektörü	Fiziksel Anlamı
$\boldsymbol{M}_0$	$[1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T$	Gezgin robot S_{MR} üzerinde (*Adım 1*)
$\boldsymbol{M}_1$	$[0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T$	Gezgin robot G_{MR}'ye ilerliyor (*Adım 1*)
$\boldsymbol{M}_2$	$[0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1]^T$	Gezgin robot G_{MR} üzerinde (*Adım 1.1* veya *2.1*)
$\boldsymbol{M}_3$	$[0\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0]^T$	Gezgin robot engel algıladı ve duruyor (*Adım 1.2*)
$\boldsymbol{M}_4$	$[0\ 0\ 1\ 0\ 1\ 0\ 0\ 0\ 0\ 0]^T$	Gezgin robot duruyor ve H_j noktası atadı (*Adım 1.2*)
$\boldsymbol{M}_5$	$[0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0]^T$	Gezgin robot engelin sınırlarını takip ediyor (*Adım 2*)
$\boldsymbol{M}_6$	$[0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0]^T$	Gezgin robot belirlediği H_j noktasına döndü ve duruyor (*Adım 2.2*)
$\boldsymbol{M}_7$	$[0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0]^T$	Gezgin robot belirlediği L_j noktasına gidiyor (*Adım 3* ve *3.1*)
$\boldsymbol{M}_8$	$[0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 0]^T$	Gezgin robot belirlediği L_j noktasına döndü ve duruyor (*Adım 3.2*)



Şekil 5. Bug-1 algoritması Petri Ağı modelinin ulaşılabilirlik ağacı (durumlar arası ilişki).

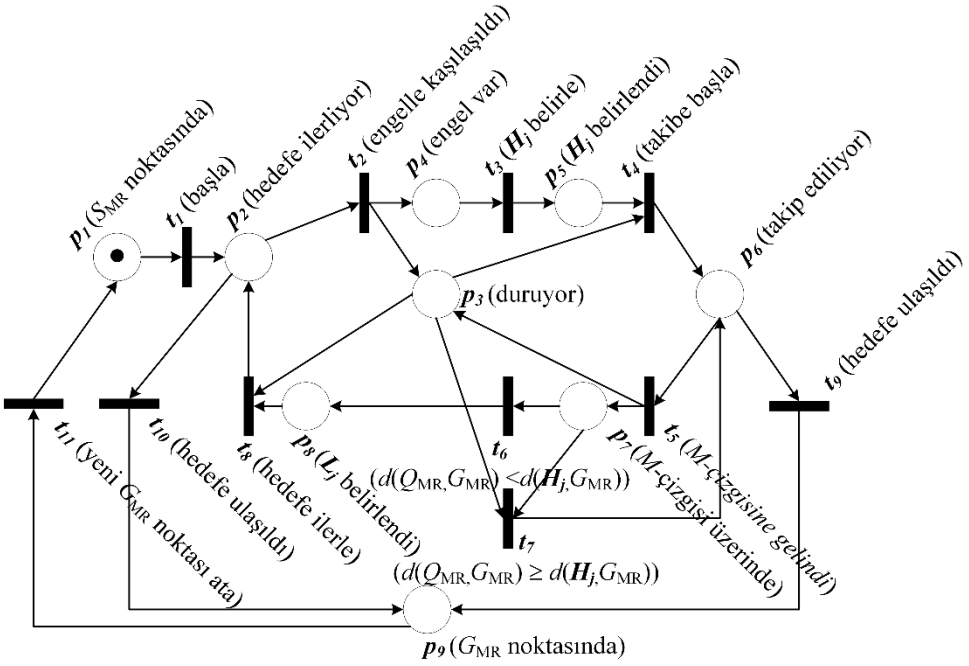
Gezgin robotun Bug-1 algoritmasını icra ederken ki birkaç durumu Petri Ağı modeli üzerinde Şekil 6'da örneklenmiştir.



Şekil 6. Benzetim ortamında Bug-1 için gezgin robotun izlediği güzergâhlar ve Petri Ağı modelindeki karşılıkları: (a) S_{MR} noktasında; (b) H_j noktasında; (c) G_{MR} noktasında.

4.2. BUG-2 Algoritması için Petri Ağı Modeli

Bölüm 2.3. içerisinde verilen Bug-2 algoritmasına yönelik hazırlanan Petri Ağı modeli Şekil 7'de verilmiştir. Bug-2 algoritmasının sabırsız olmasından kaynaklı olarak, Bug-2 Petri Ağı modeli Bug-1 Petri Ağı modeline nazaran daha az sayıda yere sahiptir.

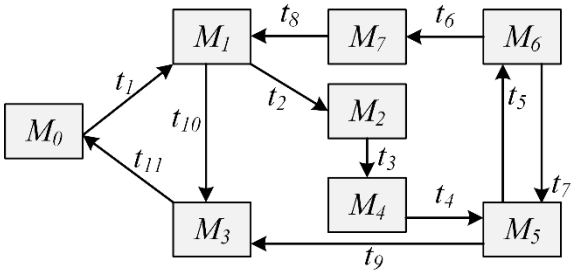


Şekil 7. Bug-2 algoritmasına yönelik hazırlanan Petri Ağı modeli

Bu model içerisinde tanımlanan Petri Ağında yerlerin kümesi $P := \{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9\}$ ve geçişlerin kümesi $T := \{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}, t_{11}\}$'dur. $\{p_1, p_2, p_3, p_4, p_5, p_6\}$ yerlerine ve $\{t_1, t_2, t_3, t_4, t_8, t_9, t_{10}, t_{11}\}$ geçişlerine ait fiziksel anlamlar Bug-1 içerisinde verilen model ile aynıdır. Farklılık olarak, p_7 yeri, gezgin robotun M-çizgisi üzerindeki bir noktada olduğunu gösterir. p_8 yeri, L_j ayrılış noktasının belirlendiğini gösterir. p_9 yeri, Bug-1 içerisinde sözü edilen p_{10} yerinde olduğu gibi gezgin robotun G_{MR} noktasına vardığını gösterir. t_5 geçişi, M-çizgisi üzerindeki bir noktaya gelindiğinin olaydır. t_6 geçişi, bir karar olayı olup $d(Q_{MR}, G_{MR})$ ve $d(H_j, G_{MR})$ Öklid uzaklıkları arasındaki $d(Q_{MR}, G_{MR}) < d(H_j, G_{MR})$ ilişkisine bakar. t_7 geçişi, bir karar olayı olup $d(Q_{MR}, G_{MR})$ ve $d(H_j, G_{MR})$ Öklid uzaklıkları arasındaki $d(Q_{MR}, G_{MR}) \geq d(H_j, G_{MR})$ ilişkisine bakar. Bug-1 modelinde bahsedildiği gibi geçişlerde sözü edilen konuma yönelik olaylar gezgin robotun gerçek zamanlı konumlama sistemi ile tetiklenmektedir. Ayrıca engel algılamaya yönelik olaylar sonar algılayıcılar tarafından tetiklenmektedir. Gezgin robot başlangıçta S_{MR} noktasındadır. Bu durum $\boldsymbol{M}_0 = [1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T$ işaretleme vektörü ile ifade edilmektedir. Bug-2 için oluşturulan Petri Ağının ulaşılabilirlik kümesi 8 adet durum içermektedir. Ulaşılabilirlik kümesi $R = \{\boldsymbol{M}_0, \boldsymbol{M}_1, \ldots, \boldsymbol{M}_7\}$ içerisindeki durumların karşılıkları ve fiziksel anlamları Tablo 2'de verilmiştir. Durumlar arası ilişkiyi gösteren ulaşılabilirlik ağacı da Şekil 8'de gösterilmiştir.

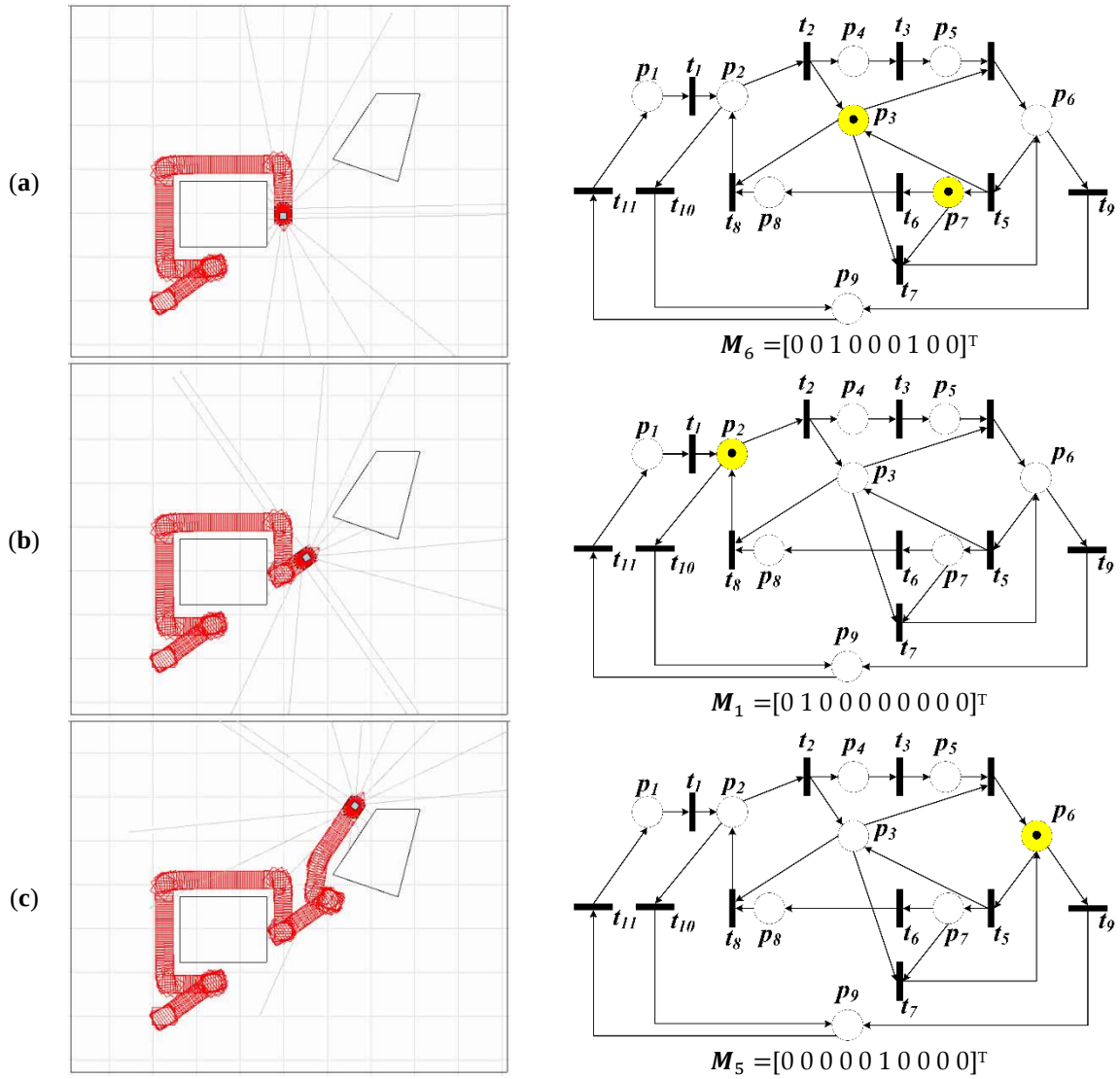
Tablo 2. Bug-2 algoritması Petri Ağı modeli için gezgin robotun tüm durumları

Durum Etiketi	İşaretleme Vektörü	Fiziksel Anlamı
$\boldsymbol{M}_0$	$[1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T$	Gezgin robot S_{MR} üzerinde (*Adım 1*)
$\boldsymbol{M}_1$	$[0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T$	Gezgin robot G_{MR}'ye ilerliyor (*Adım 1*)
$\boldsymbol{M}_2$	$[0\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 0]^T$	Gezgin robot engel algıladı ve duruyor (*Adım 1.2*)
$\boldsymbol{M}_3$	$[0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1]^T$	Gezgin robot G_{MR} üzerinde (*Adım 1.1* veya *2.1*)
$\boldsymbol{M}_4$	$[0\ 0\ 1\ 0\ 1\ 0\ 0\ 0\ 0]^T$	Gezgin robot duruyor ve H_j noktası atadı (*Adım 1.2*)
$\boldsymbol{M}_5$	$[0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0]^T$	Gezgin robot engelin sınırlarını takip ediyor (*Adım 2*)
$\boldsymbol{M}_6$	$[0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0]^T$	Gezgin robot duruyor ve M-çizgisi üzerinde (*Adım 2.2* ve *2.3*)
$\boldsymbol{M}_7$	$[0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0]^T$	Gezgin robot duruyor ve L_j noktası atadı (*Adım 3*)



Şekil 8. Bug-2 algoritması Petri Ağı modelinin ulaşılabilirlik ağacı (durumlar arası ilişki).

Gezgin robotun Bug-2 algoritmasını icra ederken ki birkaç durumu Petri Ağı modeli üzerinde Şekil 9'da örneklenmiştir.



Şekil 9. Benzetim ortamında Bug-2 için gezgin robotun izlediği güzergâhlar ve Petri Ağı modelindeki karşılıkları: (a) M-çizigisinde; (b) M-çizigisi üzerinde ilerliyor; (c) Sınır takibi.

5. SONUÇ

Bu çalışmada öncelikle 2B bir ortam içerisinde bilinmeyen engellerin bulunduğu, gidilecek hedefin koordinatlarının bilindiği ve bu hedefin bir başlangıç noktasından ulaşılabilir olduğu bir konfigürasyon içerisinde güzergâh planlaması yapılmasına yönelik geliştirilen Bug-0, Bug-1 ve Bug-2 algoritmaları tanıtılmıştır. Sözü edilen algoritmaların işleyişinin gözlemlenmesi uygulanacakları sistemler üzerindeki etkilerinin anlaşılması ve yorumlanması açısından önemlidir. Bu algoritmalar kesikli olay sistemlerinin modellenmesi ve analizinde sıklıkla kullanılan Petri Ağı ile modellenmiştir. Petri Ağı grafiksel ve matematiksel yönüyle algoritmaların işleyişinin izlenmesinde ve öngörülmesinde büyük kolaylık sağlamaktadır.

KAYNAKÇA

[1] Choset H, Lynch KM ve Hutchinson S. Principles of Robot Motion: Theory, Algorithms, and Implementations. ABD: M.I.T. Press, 2005.

[2] Yufka A ve Parlaktuna O. Performance Comparison of the BUG Algorithms for Mobile Robots. 5. "Advanced Technologies" Uluslararası Sempozyumu; 13–15 Mayıs 2009; Karabük, Türkiye. pp. 61-65.

[3] Yufka A ve Parlaktuna O. Performance Comparison of the BUG's Algorithms for Mobile Robots. "INnovations in Intelligent Systems and Applications" Uluslararası Sempozyumu; 29 Haziran–1 Temmuz 2009; Trabzon, Türkiye. pp. 111-146.

[4] Lumelsky VJ ve Stepanov A. Dynamic path planning for a mobile automaton with limited information on the environment. IEEE Transaction Automatic Control 1986; 31: 1058-1063.

[5] Sankaranarayanan A ve Vidyasagar M. A New Path Planning Algorithm for Moving a Point Object Amidst Unknown Obstacles in a Plane. "Robotics and Automation" Uluslararası Konferansı; 13–18 Mayıs 1990; Cincinnati, OH, A.B.D.: IEEE. pp. 3:1930–1936.

[6] Cassandras CG ve Lafortune S. Introduction to Discrete Event Systems, Second Edition. US, ABD: Springer, 2008.

[7] Zhou M ve DiCesare F. Petri net Synthesis for Discrete Event Control of Manufacturing Systems. Norwell, MA, A.B.D.: Kluwer Academic, 1993.

[8] Yufka A ve Aybar A. BUG Algorithm Analysis using Petri net. 8'inci "Electrical and Electronics Engineering" Uluslararası Konferansı; 28–30 Kasım 2013; Bursa, Türkiye. IEEE. pp. 507-511.

[9] Abelson H ve Disessa A. Turtle Geometry. Cambridge, MA, A.B.D.: M.I.T. Press, 1980.