Fabrikalardaki Otonom Taşıyıcı Araçlar için Çakışmasız Statik Rota Planlaması

Özge Aslan

YÜKSEK LİSANS TEZİ

Bilgisayar Mühendisliği Anabilim Dalı

Temmuz 2020

Conflict-Free Static Route Planning for Autonomous Transport Vehicle in Factories

Özge Aslan

MASTER OF SCIENCE THESIS

Department of Computer Engineering

July 2020

Fabrikalardaki Otonom Taşıyıcı Araçlar için Çakışmasız Statik Rota Planlaması

Özge Aslan

Eskişehir Osmangazi Üniversitesi
Fen Bilimleri Enstitüsü
Lisansüstü Yönetmeliği Uyarınca
Bilgisayar Mühendisliği Anabilim Dalı
Bilgisayar Yazılımı Bilim Dalında
YÜKSEK LİSANS TEZİ
Olarak Hazırlanmıştır

Danışman: Doç. Dr. Ahmet Yazıcı

Bu Tez TÜBİTAK tarafından "116E731" no'lu proje çerçevesinde desteklenmiştir.

Temmuz 2020

ETİK BEYAN

Eskişehir Osmangazi Üniversitesi Fen Bilimleri Enstitüsü tez yazım kılavuzuna göre, Doç. Dr. Ahmet Yazıcı danışmanlığında hazırlamış olduğum "Fabrikalardaki Otonom Taşıyıcı Araçlar için Çakışmasız Statik Rota Planlaması" başlıklı YÜKSEK LİSANS tezimin özgün bir çalışma olduğunu; tez çalışmamın tüm aşamalarında bilimsel etik ilke ve kurallara uygun davrandığımı; tezimde verdiğim bilgileri, verileri akademik ve bilimsel etik ilke ve kurallara uygun olarak elde ettiğimi; tez çalışmamda yararlandığım eserlerin tümüne atıf yaptığımı ve kaynak gösterdiğimi ve bilgi, belge ve sonuçları bilimsel etik ilke ve kurallara göre sunduğumu beyan ederim. 22/07/2020

Özge Aslan

İmza

ÖZET

Gelişen teknoloji ile fabrika ortamlarındaki iç lojistik taşıma görev planlama işlemi geliştirilen algoritmalar aracılığıyla yapılmaya başlamıştır. Bu alanda otomatik yönlendirmeli araçlar (OYA) için çakışmasız rota planlamaya yönelik farklı yaklaşımlar ortaya konmuştur. Endüstri 4.0 ile birlikte belli noktalar arası sabit güzergâhlar üzerinden OYA'lar tarafından yapılmakta olan taşıma görevinin, otonom taşıyıcı araçlar (OTA) tarafından yapılması öngörülmektedir. OTA'lar OYA'lardan farklı olarak belli bir şerit sistemi takip etmediği için hareket alanı daha esnektir. Taşıma görevi için OTA'ların fabrika içerisinde gidebileceği mümkün olan en uygun rotalarının belirlenmesi maliyet ve verimlilik açısından oldukça önemlidir. Aynı ortamda çok sayıda OTA'nın hareket etmesi durumunda, birden fazla OTA'nın aynı zaman aralığında aynı noktada bulunması çakışma oluşturabilmektedir. Çakışmalar istenmeyen gecikmelere neden olmakta olup bunların planlama aşamasında tespit edilmesi veya çakışma olmayacak şekilde planlanması önemlidir.

Bu tez çalışması ile çoklu OTA'lar için çakışmasız rota planlaması önerilmektedir. Çalışmada çoklu otonom taşıyıcı araçlarda çakışma problemlerini çözmek için A* algoritması temelli yeni bir algoritma önerilmiştir. Klasik A* algoritmasında aynı anda tek bir başlangıç ve bitiş düğümü için rota planlanırken, önerilen algoritmada çoklu başlangıç ve bitiş düğümleri için çakışmasız rotalar planlanabilmektedir. Algoritma çalışması sırasında olası çakışma tipleri tespit edilip, duruma uygun şekilde çözülerek rotalar oluşturulabilmektedir. Önerilen algoritma farklı hızlara sahip olan çoklu otonom taşıyıcı araçlar için test edilmiştir. Yapılan testlerde farklı çakışma tiplerine ait farklı denemeler yapılmıştır. Önerilen algoritma meydana gelen bu çakışmaları tespit ederek duruma uygun çözüm stratejisini seçip çakışmasız rotalar hesaplayabilmiştir. Önerilen algoritma fabrika içinde kullanılan OYA ve forklift gibi araçlar için de kullanılabilir.

Anahtar Kelimeler: Çakışmasız rota planlama, A* algoritması, zaman penceresi, çoklu otonom taşıyıcı araç.

SUMMARY

Task planning for the internal logistics transportation started to be carried out with the developed algorithms. In this area, different approaches have been introduced for planning conflict-free routes for automated guided vehicles (AGV). With the Industry 4.0, the transportation task, which is carried out by AGVs on fixed routes, is foreseen to be performed by autonomous transport vehicles (ATV) between certain points. ATVs are more flexible than AGVs, since they do not follow a particular lane system. It is very important in terms of cost and efficiency to determine the most suitable routes that ATVs can go within the factory for the transportation task. If many ATVs move in the same environment, they may create a conflict if more than one are located at the same point in the same time interval. Conflicts cause unwanted delays, and it is important to identify them at the planning stage and plan conflict-free routes.

This thesis proposes a conflict- free route planning for multiple ATVs. In the study, a new algorithm based on A * algorithm has been proposed to solve conflict problems in multiple ATVs. In the classical A * algorithm, while route is planned for a single start and end node at the same time, conflict free routes can be planned for multiple start and end nodes in the proposed algorithm. The algorithm can detect possible conflict types and produce nonconflict route according to the situation. The proposed algorithm has been tested for multiple ATVs with different speeds. In the tests different conflict types are generated. The proposed algorithm has been able to identify these conflicts and can select the appropriate solution strategy and generate conflict-free routes. The proposed algorithm could also be used for vehicles such as AGV, forklift that are used within the factory.

Keywords: Conflict-free route planning, A* algorithm, time window, multi-autonomous transport vehicle.

TEŞEKKÜR

Yüksek lisans eğitimim ve tez çalışmam boyunca yardımları, yönlendirmeleri ve katkılarıyla beni destekleyen değerli hocam ve tez danışmanım Sayın Doç. Dr. Ahmet YAZICI'ya sonsuz teşekkürlerimi sunuyorum. Bu süreçte karşıma çıkan her sorunda bana yardımcı olan, çözüm yolları öneren Erzincan Binali Yıldırım Üniversitesi Bilgisayar Mühendisliği bölümünde görev yapan değerli hocalarıma ve Mühendislik Fakültesinde beraber çalıştığım kıymetli çalışma arkadaşlarıma çok teşekkür ederim. Tüm hayatım boyunca yanımda olan çok değerli sevgili aileme desteklerinden dolayı minnettarım.

Bu çalışma, Türkiye Bilimsel ve Teknolojik Araştırma Kurumu'nun (TUBİTAK) 116E731 nolu "Akıllı fabrikalar için otonom taşıyıcılar ve gerekli insan-makine ve makine-makine arayüzlerinin geliştirilmesi" projesi tarafından desteklenmiştir.

İÇİNDEKİLER

Sayfa

ŞEKİLLER DİZİNİ

Şekil **Sayfa**

ÇİZELGELER DİZİNİ

ÇİZELGELER DİZİNİ (devam)

1. GİRİŞ VE AMAÇ

Görev planlama, iç lojistik yönetimi için maliyet ve zaman açısından anahtar rol oynamaktadır. Zaman içerisinde teknolojinin artan oranda fabrika ortamlarında kullanılmasıyla birlikte, rota planlamada yeni ihtiyaçlar ortaya çıkmıştır. Özellikle Endüstri 4.0 ile beraber otonom taşıyıcı araçlar (OTA) fabrikalarda görev almaya başlamış olup bunlar için çoklu rota planlama gereksinimi ortaya çıkmıştır. OTA'lar OYA'dan farklı olarak istenilen hedefe ilerlerken belli bir şerit ya da bant sistemini takip etmek yerine bulunduğu konumu ve çevresini algılayarak bu doğrultuda istenilen hedefe hareket edebilir. Dolayısıyla OTA'ların hareket alanı OYA'ların aksine daha esnek ve verimlidir. Fabrika ortamındaki taşıma görevi OTA'nın hangi rota üzerinden götürüleceğini kapsamaktadır. Yük taşıma işlemlerinin OTA'lar aracılığıyla yapılmaya başlanmasıyla birlikte, maliyet, zaman ve verimlilik vb. açısından fabrikalar önemli oranda fayda elde etmiştir. Fakat bu durumda iki problem ortaya çıkmaktadır. Bunlar; OTA'lara görevlerin (taşınacak yüklerin) en uygun şekilde atanması ve görev ataması yapılan OTA'ların en uygun rotalar üzerinden hedefe yönlendirilmesidir. OTA'lar eş zamanlı olarak hareket ettirilirken, aynı zaman diliminde aynı yerde bulunmaları durumunda çakışma adı verilen olay meydana gelmektedir. Yük taşıma işleminin belirlenen çakışmasız rotalar üzerinden gerçekleştirilmesi zaman, maliyet vb. açısından büyük oranda kazanç sağlamaktadır.

Literatürde çakışmasız rota planlamada çakışma tipleri, kafa kafaya düğüm çakışması, kafa kafaya kenar çakışması, arka arkaya düğüm çakışması vb. tiplerde sınıflandırılabilmektedir (Jia vd., 2017; Zhang vd., 2017). Meydana gelebilecek çakışma durumlarının tespiti zaman penceresi yöntemiyle yapılmaktadır (Kim ve Tanchoco, 1991). Çakışmasız rota planlama üzerine oluşturulan algoritmalar hem akademik açıdan hem de ticari açıdan oldukça önemlidir. Literatürde var olan çakışmasız rota planlama algoritmaları kesin (exact), sezgisel (heuristic), meta-sezgisel (meta-heuristic) olmak üzere üç grupta toplanabilir. Çakışmasız rota planlama çözüm yaklaşımlarından olan kesin algoritmalar genellikle çözüm zamanının düşük olacağı küçük boyutlu ortamlar üzerinde uygulanmaktadır. Fakat problem boyutu arttıkça kesin çözüm veren algoritmaların uygulanması zaman alır. Sezgisel algoritmalar ise arama uzayında daha sınırlı bir alanı dolaşır ve makul hesaplama zamanında uygun çözümler üretir. Bu sebeple çakışmasız rotalama problemi çözmek için genellikle sezgisel yaklaşımlar tercih edilmektedir. Sezgisel

algoritmalar yakınsama özelliğine sahip olmakla kesin çözümü garanti etmezler, kesin çözümün yakınındaki bir çözümü garanti ederler (Alpalsan, 2015; Ercan ve Gencer, 2013). Literatürde metasezgisel algoritmaları kullanarak çakışmasız rotalar hesaplayan çalışmalar mevcuttur (Hussein vd., 2012; Umar vd., 2013; Xidias vd., 2016). Sütun türetme yöntemi temelli çalışmalar da çakışmasız rotalamada başarılı sonuçlar vermiştir (Desaulniers vd., 2003; Krishnamurthy vd., 1993; Langevin vd., 1996). Petri ağı yöntemi de çakışmasız rota hesaplamada kullanılmıştır (Herrero ve Mart, 2010; Nishi ve Tanaka, 2012). Tamsayılı programlama kullanılarak çakışmasız rotalar oluşturabilen çalışmalar literatürde yer almaktadır (Miyamoto ve Inoue, 2016; Murakami, 2020). Çakışmasız rotalama dinamik ve statik ortam üzerine uygulanabilir (Möhring vd., 2005; Oboth vd., 1999; Smolic-rocak vd., 2010; ter Mors, 2011).

Bu tez kapsamında başlangıç bitiş noktası arası çoklu araç çakışmasız rotalama problemi çalışılmıştır. Bu kapsamda çizge şeklinde ifade edilen fabrika ortamındaki heterojen çoklu OTA'lar için çakışmasız uygun rotalar hesaplama üzerine çalışılmıştır. Çizge üzerinde yol arama algoritmalarından olan Dijksta algoritması, belli bir başlangıç ve bitiş noktası için çizge üzerindeki en kısa yolu bulmak için kullanılmaktadır. Ancak en kısa yol bulma işlemi çizge üzerindeki düğüm ve kenar sayısına bağlı olarak çok fazla zaman alabilmektedir. A* algoritması ise Dijkstra algoritmasına sezgisel uzaklık maliyetinin eklenmiş hali gibi düşünülebilir. Sezgisel fonksiyon sayesinde, A* algoritması tüm düğümleri dolaşmak yerine hedefe gidebileceği düğümleri dolaşır. Literatürde Dijkstra ve A* algoritmasını veya bu algoritmaların geliştirilmiş versiyonlarını kullanarak çakışmasız rotalar hesaplayan çalışmalar mevcuttur (Chen vd., 2013; Cui vd., 2019; Lin vd., 2017; Maza ve Castagna, 2005; Qing vd., 2017; Zhang vd., 2019). Tez çalışması kapsamında heterojen çoklu OTA'lar için çakışmasız rota planlama algoritması geliştirilmiştir. Çakışmasız uygun rotaları hesaplamak için literatürdeki klasik A* algoritması revize edilerek NC-A* (NonConflict-A*) algoritması önerilmiştir. Önerilen algoritma birden fazla başlangıç ve bitiş düğümü için eş zamanlı olarak uygun rotaları hesaplayabilmektedir. Geliştirilen NC-A* algoritmasında klasik A* algoritmasındaki açık ve kapalı liste haricinde gidilen veya gidilmesi muhtemel kenarlarda bir liste veri yapısı halinde tutulmaktadır. Meydana gelebilecek kenar çakışmaları bu liste üzerinden, düğüm çakışmaları kapalı liste üzerinden kontrol edilmektedir. Tespit edilen çakışmalar duruma

uygun olarak düğümde bekletme, kenarda bekletme veya rota değiştirme şeklinde çözümlenmiştir.

Tezin takip eden bölümünde görev planlama üzerine literatürde yapılan çalışmalar gruplanarak özet halinde verilmektedir. Bölüm 3'de ise çakışma tipleri ayrıntılı bir şekilde açıklanarak önerilen yöntem detaylı bir şekilde verilmektedir. Son olarak tez çalışması kapsamında kullanılan ortam tanıtılmış ve çakışma probleminin nasıl çözüleceği örnekler üzerinde açıklanmıştır. Bölüm 4'de ise yapılan testler ve sonuçları paylaşılmıştır. Son bölümde ise edilen sonuçlar değerlendirilmiş ve gelecek çalışmalar için bilgiler verilmiştir.

2. LİTERATÜR ARAŞTIRMASI

Malzeme taşıma görevini maliyet ve zaman açısından daha verimli yapan OTA'lar için görev ve rota planlama konuları önemli araştırma alanları olarak görülmektedir. Rota planlamada çakışmasız rotaların planlanması trafik yönetimi için önemli bir yere sahiptir. İç lojistikte taşıma talep katakteristiğine bağlı olarak farklı rotalama algoritmalarının geliştirilmesine ihtiyaç vardır. Topla dağıt rotalama veya başlangıç bitiş rotalama yapılabilir. Topla dağıt rotalamada belirlenen taleplerin durumuna göre birden fazla nokta ziyaret edilir ve uygun rotalar hesaplanırken taleplerin olduğu bu noktalara uğramak zorunludur. Başlangıç bitiş rotalamada ise belirlenen iki nokta arasındaki en uygun rota hesaplanır ve başlangıç bitiş noktaları dışında herhangi bir noktaya uğrama zorunluluğu yoktur.

2.1. Çakışmasız Rota Planlama

Aynı ortamda hareket eden iki veya daha fazla OTA aynı zaman diliminde aynı düğümü veya kenarı kullanmak istediğinde çakışma durumu meydana gelmektedir. Literatüre bakıldığında çakışma tipleri: i) literatürde kafa kafaya düğüm çakışması (head-on node conflict) olarak geçen aynı doğrultuda düğüm üzerinde karşılıklı çakışma, ii) literatürde genelde düğüm çakışması veya çapraz çakışma olarak geçen (node/cross conflict) farklı doğrultuda düğüm üzerinde karşılıklı çakışma, iii) literatürde arka arkaya düğüm çakışması (catch-up node conflict) olarak geçen hız farkından kaynaklanan arka arkaya düğüm çakışması, iv) literatürde arka arkaya kenar çakışması (catch-up edge conflict) olarak geçen hız farkından kaynaklanan arka arkaya kenar çakışması, v) literatürde kafa kafaya kenar çakışması (head-on node conflict) olarak geçen kenar üzerinde karşılıklı çakışma olmak üzere 5 grupta incelenebilir (Liu vd., 2017; Zhang vd., 2018).

Literatürde fabrikalardaki taşıyıcı araçlar için çakışmasız rotalar hesaplanırken farklı algoritmalar ve yaklaşımlar kullanılmıştır. Metasezgisel algoritmalar çakışmasız rotalar hesaplanırken kullanılan yöntemlerden biridir. Hussein vd. (2012) yaptıkları çalışmada, rota planlama problemi için yörünge tabanlı metasezgisel yöntemler ile popülasyon temelli metasezgisel birkaç yöntemi karşılaştırmışlardır. Derin öncelikli arama (Breadth-first deterministic search) yöntemi ile bulunan rotaların optimum kabul edildiği çalışmada,

yasaklı arama (tabu search), tavlama benzetimi (simulated annealing) ve genetik algoritma yöntemleri birbirleriyle karşılaştırmışlardır. Buna göre hesaplama zamanı açısından tavlama benzetimi algoritması ön plana çıkarken, en kısa rotayı hesaplamada yasaklı arama algoritması en başarılı olmuştur. Umar vd. (2013) OYA'ların çakışmasız rotalanması için görev tamamlanma süresini ve toplam iş gecikmesini en aza indirgeyen çok amaçlı genetik algoritma yaklaşımını önerdikleri çalışmalarını yayınlamışlardır. Çalışmada önerilen genetik algoritmadaki kromozom yapısının bir bölümü OYA'lara görev atamasını temsil ederken kalan bölümü OYA'ların rotalanmasını temsil etmektedir. Çalışma, rotalama için önceliğe dayalı genetik algoritmayı ve görev atama için rastgele anahtar tabanlı temsili (random-key representation) seçmektedir. OYA'lar için hesaplanan rotalar üzerinde çakışma meydana geldiğinde, rota popülasyondaki diğer çözümle yer değiştirmiştir. Çalışmanın öne çıkan kısmı iki farklı amaç fonksiyonunu birleştirmiş olmasıdır.

Xidias vd. (2016) otonom araç filolarının rotalanması ve çizelgelenmesi üzerine çalışmışlardır. Önerilen yaklaşımda iki boyutlu ortam Bump-Surface konseptiyle temsil edilmiştir. Düşük maliyetli ve çakışmasız en uygun rotaların bulunması için genetik algoritmayı değiştirerek kullanmışlardır. Çalışmada otonom araçların aynı depoyu paylaşması durumunda çakışma olmaması için bekleme stratejisi uygulanmıştır. Dört farklı ortam üzerinde test edilen yöntemin başarılı şekilde rotalar oluşturduğu gösterilmiştir. Tsang vd. (2018) depo yönetiminde homojen robotlar için çakışmasız yol planlama algoritması ve görev tahsisi konularına çalışmışlardır. Önerilen sistemde görev tahsisi merkezi sunucular tarafından gerçeklenir. Önerilen algoritma her robot için yapay potansiyel alan yöntemi kullanılarak yerel rotalar hesaplayarak çalışmaya başlar. Yerel rota planlama da düşük hesaplama yükü olan rota, çevrimiçi planlama için robotun hemen veya yakın zamandaki ortamını dikkate alır. Robot sunucudan görev tahsisini aldığında yakınlık algılayıcıları tarafından alınan ortam bilgisine dayanarak dağıtımı sağlayacak şekilde yol planlaması yapar. Robot hedefine ulaştığında görev tahsisi performansını iyileştirmek için merkezi sunucuya kat ettiği mesafeyi geri bildirir. Çalışmada görev tahsisi algoritması olarak genetik algoritma kullanılmıştır.

Çakışmasız rotalar hesaplanırken kullanılan yöntemlerden biri sütun türetme (column generation) yöntemidir. Krishnamurthy vd. (1993) yaptıkları çalışmada, OYA rotalama probleminin statik versiyonunu geliştirmişlerdir. Burada çift yönlü trafik ağı içeren

bir ortamda çakışmasız rota planlama için tamamlanma zamanını en aza indirgemeyi amaçlayan sütun türetme algoritmasını geliştirmişlerdir. Desaulniers vd. (2003) yaptıkları çalışmada, OYA'ların eşzamanlı olarak görev ataması ve çakışmasız rotalanması sorununu çözmeyi amaçlayan bir algoritma önermişlerdir. Sütun türetme ve dal-kesme (Branch-Cut) yöntemleri temeline dayanan algoritma, gecikme maliyetlerini en aza indirgemeyi amaçlamaktadır. Algoritma, (Krishnamurthy vd., 1993; Langevin vd., 1996) çalışmalarına benzemekle birlikte, (Krishnamurthy vd., 1993)'den farklı olarak OYA'lara görev atama problemini de ele almıştır. (Langevin vd., 1996) çalışmasından farkı ise ikiden fazla OYA için çözüm üretebilmesidir.

Çoklu OYA'lar için çakışmasız uygun rotalar oluşturmada kullanılan yöntemlerden biri Petri ağı yöntemidir. Herrero ve Mart (2010) yayınladıkları makalede esnek üretim sistemindeki çoklu OYA'lar için merkezi olmayan navigasyon kontrolünü kullanarak çakışmaları önleyen bir yöntem önermişlerdir. Önerilen yöntemde görev atamalarını ve trafik kontrol problemlerini çözmek için ise dağıtık Petri ağı kullanılmıştır. Petri ağı kullanan diğer bir çalışmada ise Nishi ve Tanaka (2012) çift yönlü trafik akışına sahip ortamdaki OYA'lar için eşzamanlı görev ataması ve çakışmasız rotalama problemi için Petri ağı ayrıştırma yaklaşımı önermiştir. Eş zamanlı görev atama ve rotalama problemi, Petri ağında optimum geçiş ateşleme problemi şeklinde tanımlanmıştır. Bununla beraber Petri ağı ayrışma yaklaşımını karmaşıklığı da azaltmıştır.

Çakışmasız rotalama problemini tamsayılı programlama ile formüle eden çalışmalar mevcuttur. Miyamoto ve Inoue (2016) kapasite kısıtlı OYA'ların görev atama ve çakışmasız rotalaması üzerine çalışmışlardır. Makalede rota bulma problemi tamsayılı programlama ile formüle edilmiştir. Probleme çözüm olarak yerel ve rasgele olmak üzere iki farklı arama metodu önerilmiştir. Yapılan testler sonucunda karşılaştırılan bu iki metoddan yerel arama metodu daha hızlı çözüm bulması sebebiyle öne çıkmıştır. Murakami'nin (2020) yaptığı makale, kapasite kısıtlı OYA sisteminin görev atama ve çakışmasız rotalama problemi olarak bilinen bir OYA rotalama problemini ele almaktadır. Problemi modellemek ve karma-tamsayılı doğrusal programlama problemi olarak formüle etmek için bir zaman-uzay ağı (Time-Space Network) kullanılmıştır. Zaman-uzay ağında OYA'ların ve yüklerin akışları ayrı ayrı ele alınmıştır ve daha sonra bu akışlar formülasyondaki kısıtlamaları göz önünde bulundurularak senkronize edilmiştir. Bu yaklaşım, problemin bir karma-tamsayılı doğrusal

programlama problemi olarak formüle edilmesini sağlar. Deneysel sonuçlar, önerilen yaklaşımın incelenen örneklerin çoğunda en uygun çözümleri bulabileceğini göstermektedir.

Çakışmasız rota oluşturma problemi dinamik veya statik ortam üzerinde çözülebilir. Oboth vd. (1999) yaptıkları çalışmada, (Krishnamurthy vd., 1993) çalışmasındaki statik ortam için geliştirilen çözüm metodojilerini, dinamik ortamda gerçekleştirmeyi amaçlamışlardır. K tane OYA için taleplerin tamamının karşılanma süresini en aza indirgeyecek şekilde çakışmasız rota bulmak amaçlanmıştır. Rota planlama işlemi en kısa görev tamamlanma zamanına sahip OYA'dan başlayarak sırayla yapılmaktadır. Rota planlama esnasında her OYA için kendinden önce rota planlaması yapılan tüm OYA'ları dikkate alarak çakışmasız rota planlaması yapılmaktadır. Çalışmada bununla beraber görevi tamamlanan OYA'lar için uygun konumlandırma yöntemleri de açıklanmıştır. Möhring vd. (2005) çalışmalarında otomatik lojistik sistemlerinde OYA'ların çakışmasız rotalanması için bir algoritma sunmuşlardır. Algoritma, her istek için gerçek zamanlı hesaplama, zaman pencereleri ile en kısa yolun belirlenmesini ve bu zaman pencerelerinin yeniden düzenlemesini içermektedir. Hamburg Limanı Konteyner Terminal Altenwerder (CTA) 'da kullanılan statik rotalama yaklaşımı ile karşılaştırıldığında, algoritmanın statik olan yaklaşımdan daha üstün olduğu görülmüştür.

Smolic-rocak vd. (2010) yaptıkları çalışmada çoklu OYA'ların çakışmasız rotalaması için dinamik yönlendirme yöntemini önermişlerdir. Rotalama problemini dinamik olarak çözmek için zaman penceresi yöntemini kullanmışlardır. Her OYA için hesaplanan rotaların uygunluğu zaman penceresi vektörü üzerinden kontrol edilir. Çalışmada önerilen algoritmanın dairesel rotaya izin verirken çakışmaları önleyerek başarılı rotalar bulduğu gösterilmiştir. Mors (2011) oluşturduğu yayında düşük polinom zaman karmaşıklığında çalışabilen dinamik ortamlar için çakışmasız rota planlama algoritması önermiştir. Algoritmada rotalar robotların öncelik sırası dikkate alınarak sırasıyla her robot için ayrı ayrı oluşturulmaktadır. Robotlara ait rotalar hesaplanırken bu rotanın en uygun rota olup olmadığı Pareto optimal planı kullanılarak kontrol edilir. Li vd. (2018) yaptıkları çalışmada tek yönlü trafik akışına sahip ortamda bulunan OYA'lar için çakışmasız rota planlaması yapmışlardır. Çalışmada yalnızca arka arkaya çakışma (catch-up conflict) tipine çözüm önerilmiştir. Meydana gelebilecek çakışma durumları olasılıksal yoğunluk

fonksiyonu ile tahmin edilmiş ve dinamik ayarlama yöntemi (Dynamic adjustment method) kullanılarak önlenmiştir.

Fabrikalardaki çoklu OYA'lar için en uygun rotalar hesaplanırken aynı zamanda enerji tüketimini en aza indirgemeyi amaçlayan çalışmalar da literatürde mevcuttur. Qiu vd. (2015) yayınladıkları çalışmada heterojen OYA'lar için enerji tüketimini minimize eden rota planlaması üzerinde çalışmışlardır. Enerji tüketimini hesaba katan rotalama problemini çözmek için parçacık sürü optimizasyonu (PSO) algoritması kullanılmıştır. Çalışmada yük kapasiteleri farklı olan iki çeşit OYA üzerinden testler yapılmıştır. Çalışmanın sonucunda enerji tüketimini dikkate almayan heterojen OYA rotalama ile enerji tüketimini dikkate alan heterojen OYA rotalama karşılaştırılmış olup, deney sonucunda enerji tüketimini dikkate alan heterojen OYA rotalama ile %8,37 toplam mesafe artmasına rağmen %10,37 enerji tasarrufu sağlanmıştır. Adamo vd. (2018) yaptıkları çalışmada değişken hızla hareket eden çoklu OYA'lar için çakışmasız rota planlaması üzerine çalışmışlardır. Toplam harcanan enerjiyi en aza indirgemeyi amaçlayan çalışmada, çakışmasız rotalar dal-kesme algoritması kullanılarak oluşturulmuştur. Hesaplanan rotalar üzerindeki çakışmalar zaman pencereleri üzerinden kontrol edilmektedir. Çakışma durumunda ise dallanma prosedürü zaman ve alan karmaşıklığı daha az olan rotayı seçmektedir. Önerilen algoritma 400 düğümlü bir ortam üzerinde test edilmiş olup, kısa sürede uygun rotayı hesaplayabilmiştir.

Çakışmasız rota planlama ile ilgili çalışmalarda ortamdaki araç sayısı optimum çözüm üretmede son derece önemlidir. Çakışmasız rota planlama ile ilgili çalışmaların bazıları az sayıdaki taşıyıcı araç için optimum çözüm üretebilmektedir. Langevin vd. (1996) yaptıkları çalışmada, görev atama ve çakışmasız rotalama sorununa bir yaklaşım önermişlerdir. Önerilen yaklaşım bütün görevlerin tamamlanma zamanını en aza indirgemeyi amaçlar. Yöntem dinamik programlamaya dayanır ve iki araçlı örnekler için optimum çözümler üretebilmektedir. İkiden fazla araç için, dinamik programlamada durum sayısı çok olduğundan dolayı önerilen yöntem optimum çözüme ulaşmaz. Ancak bu durumda yöntem, her aşamada keşfedilecek durumların sayısını sınırlandırarak sezgisel bir algoritma olarakta kullanılabilir. Fazla sayıda araç içeren büyük boyutlu ortamlar için uygun çözüm üretebilen çalışmalar mevcuttur. Yan vd. (2017) tek yönlü ve çift yönlü trafik akışı içeren ortamlarda çakışma ve tıkanıklığı önlemek için bir takım kurallar dizisi önerdikleri çalışmalarını yayınlamışlardır. Çalışmada, çakışmadan kaçınma ve OYA'ların merkezle

haberleşme karmaşıklığı belirli oranda çözülmüştür. Karmaşıklık durumunda bulunan çözüm merkez tarafından tüm OYA'lara iletilerek çevrimiçi bir çözüm platformu oluşturulmuştur. Çalışmada polinom zaman karmaşıklığında çalışan kurallar dizisinin fazla sayıda OYA içeren büyük ortamlara da uygulanabileceği belirtilmiştir.

Małopolski'nin (2018) yaptığı çalışmada, çakışmasız OYA ile ilgilli önerilen algoritmaların az sayıda OYA'ya sahip sistemler için uygun olduğundan bahsedilmiştir. Yeni ulaştırma sistemlerinde ise çok sayıda OYA kullanılmaktadır. Bu makalede ulaştırma sistemlerin tek yönlü, çift yönlü veya çok şeritli olarak tanımlanması için yeni bir yöntem önerilmiştir. Ulaştırma sisteminin ortamı karelere bölünür ve bir matris ile tarif edilir. OYA hareketi sabit ortalama hız ile kareden kareye olarak kabul edilmiştir. Çarpışmalar ve kilitlenme önleme için önerilen yöntem, temel rezervasyon zincirlerine dayanmaktadır. Bu yöntem, herhangi bir zamanda birçok OYA için rezervasyonların dinamik olarak eklenmesini sağlar. Kuyruklara rezervasyon ekledikten hemen sonra, OYA görevine başlayabilir. Temel rezervasyon zincirleri sayesinde çakışmalar ya da çıkmazlar asla gerçekleşemez. Ortam, bir kare topoloji olarak tanımlandığından, tüm sıralar kolayca alt kümelere bölünebilir ve rezervasyon durumu eş zamanlı olarak güncelleştirilebilir ve analiz edilebilir. Bu olasılık, çok sayıda OYA'ya sahip ulaşım sistemleri için çok önemlidir. Thanos vd. (2019) yaptıkları çalışma kapasiteli araçların görev atama ve çakışmasız rotalama problemini, depolama tahsisi problemiyle bütünleştirmekle birlikte bir depo ortamında yük toplama, teslimat ve yer değiştirme taleplerinin toplam tamamlanma süresini en aza indirgemeyi amaçlamaktadır. Çalışmada hesaplama süresini azaltmak için çevrimdışı rota oluşturma yöntemini kullanarak, geliştirilmiş formülasyona dayalı olarak bir sezgisel çözüm önerilmiştir. Deneysel sonuçlar, önerilen yöntemin büyük ölçekli örnekler için etkinliğini göstermektedir ve iyileştirilmiş depolama alanı ayırma kararlarının hizmet sürelerini nasıl azaltabileceğini vurgulamaktadır.

Literatürde çakışmasız rotalama problemi bazı çalışmalarda çizelgeleme problemi ile birlikte ele alınmıştır. Nishi vd. (2011) yaptıkları çalışmada OYA'ların eş zamanlı çizelgeleme ve çakışmasız rotalama problemini çözmek için iki aşamalı ayrışma algoritmasını ele almışlardır. Ana problem olarak adlandırılan problemde lagrange gevşemesi ile çizelgeleme problemi çözülürken, alt problemde ise ana problemdeki uygun çözüm kullanılarak rotalama problemi çözülür. Çalışmada toplam ağırlıklı gecikmeyi en aza

indirgemek amaçlanmıştır. Giordani vd. (2013) yaptıkları çalışmada mobil robotlar için üretim planlama ve çizelgeleme üzerinde çalışmışlardır. Çalışmada önerilen çözüm iki seviyeli merkezi olmayan çok ajanlı (agent) bir yaklaşımdır. İlk aşama üretim planlama aşamasıdır ve burada ajanlar robotlar için rekabet eden görevlerdir. İkinci aşama çizelgeleme aşaması olup burada ajanlar birinci seviyeden gelen talepleri yerine getirmek için kendilerini farklı görevler arasında yeniden konumlandıran robotlardır. İlk aşamada auction based negotiation protocol kullanılırken, ikinci aşamada macar yönteminin dağıtılmış versiyonuyla çok robotlu görev tahsisi problemi çözülmüştür. Mehrabad vd. (2015) yayınladıkları çalışmada atölye tipi çizelgeleme probleminin ve çoklu OYA'ların çakışmasız rotalanması probleminin entegre modeline odaklamışlardır. Problemi çözmek için karma-tamsayılı lineer programlama (mix-integer linear programming) ve iki aşamalı karınca kolonisi (Ant Colony) algoritması önerilmiştir. 13 test problemi üzerinde karınca kolonisi algoritması ile karma-tamsayılı lineer programlama modeli karşılaştırılmıştır. Yapılan bu testler sonucunda çalışmada kullanılan yöntemin büyük boyutlu problemlerde karınca kolonisi algoritmasına göre optimal sonuçlar elde ettiği ve makul sürede rotalama problemini çözebildiği görülmüştür. Sun vd., (2018) yayınladıkları çalışmada akıllı kapalı otopark sistemlerindeki çoklu OYA'lar için dinamik zaman pencereli çizelgeleme önermişlerdir. Çalışmada rota hesaplaması Dijkstra algoritması kullanılarak yapılmıştır. Çalışmada arka arkaya çakışma (catch-up conflict), doğrusal kafa kafaya çakışma (opposite conflict) ve doğrusal olmayan kafa kafaya çakışma (intersection conflict) olmak üzere üç farklı çakışma türünden bahsedilmiştir. Arka arkaya çakışma ve doğrusal olmayan kafa kafaya çakışma için çözüm stratejisi olarak düşük öncelikli OYA'yı bekletme stratejisi önerilmiştir. Doğrusal kafa kafaya çakışma olan durumlar için yolun aynı anda iki OYA'nın geçebileceği genişlikte olduğu varsayılmıştır.

Bae ve Chung (2018) çoklu heterojen OYA'ların rota planlaması üzerine çalışma yayınlamışlardır. Çalışmada kullanılan OYA'lar yük ve hız bakımından heterojendir. Çizelgeleme ve rota planlama problemi çoklu heterojen Hamilton yolu problemi olarak ifade edilmiştir. Bu problemi çözmek için çalışmada primal-dual tekniğini önermişlerdir. Önerilen yöntem kısa hesaplama zamanı içerisinde uygun çözümler bulmuştur. Tai vd. (2019) yaptıkları çalışmada otomatik depolardaki OYA'lar için çakışmasız rota planlama yapmışlardır. Önerilen yöntem harita oluşturulması, alternatif rotaların belirlenmesi, alternatif rotalar için çizelgeleme yapılması ve en kısa maliyetli yolun seçilmesi olmak üzere

dört bölümden oluşmaktadır. Alternatif rotalar k-en kısa yol algoritması (k-shortest path algorithm) kullanılarak oluşturulmuştur. Bu alternatif rotaların zaman pencereleri oluşturularak meydana gelebilecek çakışmalar bu zaman pencereleri üzerinden çözülmüştür. Lyu vd. (2019) yaptıkları çalışma esnek bir üretim sistemindeki makine ve OYA çizelgeleme problemine odaklanmaktadır. Çalışmada optimum sayıda OYA sayısını, en kısa görev tamamlama süresini, çakışmasız rotalama problemini aynı anda çözebilen bir yöntem önerilmiştir. Çalışmada tüm bu problemleri çözmek için zaman penceresi metoduna dayanan Dijkstra algoritması ile genetik algoritma birlikte kullanılmıştır. Amaç, OYA'ların sayısının etkisini göz önüne alarak, tamamlanma zamanını en aza indirmektir. OYA'ların sayısının arttırılması, ilk aşamadaki tamamlanma zamanı üzerinde önemli bir etkiye sahiptir. Bununla birlikte, OYA'ların sayısı bir eşik değere ulaştığında, üretim istikrar kazanma eğilimindedir. Yapılan deneyin sonuçları, uygun sayıda OYA seçmenin önemini vurgulamıştır. Üretim sisteminde sabit sayıda OYA kullanılmasının, kaynakların boşa harcanmasına ve işletmeye gereksiz bir şekilde yük oluşturduğuna veya yetersiz kaynaklara yol açabildiğine değinilmiştir. Riazi vd. (2019) yayınladıkları çalışmada OYA'ların çakışmasız çizelgelemesini ve rotalamasını çözmek için Benders ayrışmasına dayanan mevcut bir yaklaşıma yeni bir sezgisellik ve çeşitli iyileştirmeler önermişlerdir. Mevcut yöntem sorunu görev atama/sıralama ve çakışma önleme kontrolü olmak üzere iki aşamada çözer ve büyük ölçekli OYA sistemleri için uygun değildir. Mevcut yöntemin önerilen sezgisel versiyonu ise büyük ölçekli OYA sistemlerinde çok daha kısa sürede iyi çözümler sağlamıştır.

Zhang vd. (2019) yaptıkları çalışmada çoklu robot görev atamaları ve rota planlama problemi için bir çözüm önermişlerdir. Çalışmada vitality-driven genetik görev atama algoritması önerilmiş olup bu algoritma görev çizelgeleme ve çakışma problemlerini çözebilmektedir. Algoritma, rastgele mutasyon gibi yerel operatörler, açgözlü çaprazlama ve canlılık seçimi durumlarını içermektedir. Önerilen algoritma memetik algoritma, balık sürü algoritması, genel değişken komşu arama yöntemleriyle karşılaştırılmıştır ve önerilen algoritmanın başarılı sonuçlar elde ettiği görülmüştür. Thomas vd. (2019) yaptıkları çalışmada gelişmiş robotlar için rota planlama ve çizelgeleme üzerine bir yöntem önermişlerdir. Extended Conflict-Based arama adını verdikleri yöntem meydana gelen çakışmaları kısıtlama ağacı (Constraint Tree)'nı kullanarak çözmüşlerdir. Meydana gelen birbirini dışlayan (mutually exclusive) çakışmaları çözebilmek için Geçici Doluluk Grafı (Temporal Occupancy Graph) kullanılmıştır. Bununla birlikte her robot meydana

gelebilecek çakışmalardan haberdar edilerekte çakışmalar önlenmektedir. Elde edilen sonuçlar önerilen algoritmanın çok sayıdaki robotlar için bile etkili rotalar bulabildiğini göstermiştir.

2.2. Başlangıç Bitiş Çakışmasız Rota Planlama Yaklaşımları

Literatürde belli başlangıç ve bitiş noktaları arasında birden çok araç için çakışmasız rota planlama yaklaşımları geliştirilmiştir. Bu alanda A* veya Dijikstra algoritmaları temelli çalışmalar vardır. Kim ve Tanchoco (1991) yaptıkları çalışmada, çift yönlü akış yolu içeren ulaşım ağındaki OYA'lar için Dijkstra algoritmasını kullanarak çakışmasız en kısa rotaları hesaplamışlardır. Çalışmada düğümlerin uygun zaman pencelerini, kenarların ise bu zaman pencerelerine erişebilirliğini ifade ettikleri bir zaman penceresi çizgesi kullanılmıştır. OYA'ların çakışmasız rota planlaması ise oluşturulan bu zaman penceresi çizgesinin uygun zaman pencereleri üzerinden yapılmaktadır. Maza ve Castagna (2001) yayınladıkları çalışmada (Kim ve Tanchoco, 1991)'daki çalışmasına gerçek zaman kontrolü durumunu ekleyerek çift yönlü trafik akışına sahip ortamda hareket eden OYA'lar için çakışmasız rota problemi üzerine bir yöntem önermişlerdir. Maza ve Castagna (2005) çalışmalarında çift yönlü OYA sistemlerinin çakışmasız rotalamasını ele almışlardır. Çalışmada iki aşamalı çakışma kontrol stratejisi önerilmiştir. İlk kontrol aşamasında, (Kim ve Tanchoco, 1991) tarafından önerilen bir ön planlama yöntemi, OYA'lar için en hızlı çakışmasız yolları oluşturmak için kullanılır. İkinci aşamanın amacı, planlanan rotalar korunurken herhangi bir kesinti durumunda kilitlenmelerin(deadlock) ortadan kaldırılmasıdır. Bunun için RVWA (Robust Vehicle Waiting Algorithm), RVRAA (Robust Vehicle Routing Ahead Algorithm) ve RVDA (Robust Vehicle Delaying Algorithm) adı verilen üç algoritma ikinci aşamada kullanılmaktadır.

Chen vd. (2013) yaptıkları çalışmada küçük boyutlu çoklu OYA ortamları için çakışmasız rota planlamak için Dijkstra algoritmasını ve dinamik zaman penceresi metodunu birlikte kullanmışlardır. Çalışmada doğrusal ve doğrusal olmayan kafa kafaya çakışma (cross/head-on conflict) olmak üzere iki farklı çakışma türüne çözüm önerilmiştir. Çalışmadaki OYA'ların hızları aynı olduğu için arka arkaya çakışma (catch-up conflict) türü meydana gelmeyeceğinden dolayı bu çalışmada ihmal edilmiştir. Yapılan test sonuçlarında önerilen yöntemin küçük boyutlu çoklu OYA ortamları için meydana gelen çakışmaları

çözdüğü görülmüştür. Qing vd. (2017) yaptıkları çalışmada otomatik depolama boşaltma sistemlerindeki karesel ortam haritası için rota planlama üzerine çalışmışlardır. Çalışmada klasik Dijkstra algoritmasının geliştirilmiş bir versiyonu önerilmiştir. Klasik Dijkstra algoritmasında yalnızca bir adet en kısa yol bulunur ve eşit mesafeye sahip birden fazla en kısa yol varsa diğer yollar dikkate alınmaz. Çalışmada önerilen algoritmada ise eşit mesafeye sahip tüm en kısa yollar kaydedilir ve bu eşit mesafeli yollardan dönme faktörü dikkate alınarak tamamlanma zamanı en kısa olan yol seçilir. Yapılan deneyler önerilen algoritmanın aynı mesafeye sahip yollar içerisinden mesafe ve zaman bakımından en optimum olanı seçtiğini göstermiştir. Zhang vd.'nin (2017) yaptıkları çalışma OYA'ların çevrimdışı rota planlaması ile ilgilidir. Çalışmada OYA'ların kontrol sistemi için bir depo sisteminde çakışmasız bir rota planlama yaklaşımı önerilmiştir. Aday rotalar Dijkstra algoritması kullanılarak belirlenmiştir. Çakışmaların çözülmesi için iki yöntem kullanılıp bunlar karşılaştırılmıştır. Bu yöntemlerden biri çakışma durumunda OYA'lardan birini bekletmek olup, diğer yöntem ise OYA'lardan birinin rotasının yeniden planlanmasıdır. Rotası hesaplanacak olan OYA kendinden önce rotası hesaplanan tüm OYA'ların rotasını dikkate alarak çakışmasız şekilde rotalanır.

Zhang vd.'nin (2018) yaptıkları çalışma çakışmasız OYA rota planlama için dört çakışma türü ve bu çakışmaları çözmek için ise üç çözüm stratejisi sunmaktadır. Çalışmada (Qing vd., 2017) çalışmasında önerilen geliştirilmiş Dijkstra algoritması kullanılarak rota hesaplaması yapılmıştır. Hesaplanan rotalarda ortaya çıkan çakışma durumunda, çözüm için üç strateji önerilmiştir. Bu stratejiler; çakışma olduğundan OYA'lardan birinin rotasını Dijkstra algoritması kullanılarak hesaplanan alternatif rota (varsa) ile değiştirmek, OYA'lardan birini çakışma durumu ortadan kalkana kadar bekletmek ve rota hesaplaması yapılan OYA'nın rotasını çakışmasız rota ile değiştirmektir. Çalışmada her bir çakışma türü için bu üç çözüm stratejisi denenmiş ve sonuçları karşılaştırılmıştır. Çakışmanın türüne göre hangi çözüm stratejisinin daha iyi sonuç verdiği de değişmektedir. Kim vd. (2019) yaptıkları çalışmada dinamik yükler altında hareket eden OYA'lar için rota planlaması yapan bir yöntem önermişlerdir. Dijkstra algoritmasının geliştirilmiş versiyonu kullanılan yöntemde OYA'ların enerji tüketimini ve toplam mesafeyi minimize etmek amaçlanmıştır. Enerji tüketiminin hesabında kullanılan ortamın fiziksel yapısı (eğim vb.) da dikkate alınmıştır. Simülasyon sonuçları, OYA'ların geliştirilen Dijkstra algoritması kullanılarak oluşturulan rota üzerinde çalıştığında, enerji tüketiminin en aza indirgenebileceğini doğrulamaktadır.

Literatürde çakışmasız rota planlaması için A* algoritmasını kullanan çalışmalarda genellikle algoritmanın maliyet fonksiyonu geliştirilmiş veya maliyet fonksiyonuna ortamdaki araçların dönme faktörü dâhil edilmiştir. Dönme faktörü, OYA'nın düz yolda giderken sahip olduğu hız ile dönme noktalarında sahip olduğu hızın aynı olmamasından kaynaklanır. Wang vd. (2015) OYA'ların en kısa rota problemini çözmek için geliştirilmiş A* algoritmasını önermişlerdir. Çalışmada en kısa rotalar hesaplanırken dönme faktörü dikkate alınmıştır. Rota uzunluğu aynı iki en kısa rota düşünüldüğünde daha fazla dönme noktasının olduğu rotada OYA daha fazla vakit kaybeder. Çalışmada çakışma durumu da hesaba katılmıştır. Düğümlere giriş zamanı ve düğümlerden ayrılış zamanlarına bakılarak aynı zaman aralığında birden fazla OYA aynı düğümde bulunduğu zaman OYA'lardan biri bekletilmiştir. Önerilen geliştirilmiş A* algoritması, Dijkstra ve klasik A* algoritmasıyla rota uzunluğu, dönme faktörü, dolaşılan düğüm sayısı bakımından karşılaştırılmıştır ve en iyi sonuçların geliştirilmiş A* algoritması ile elde edildiği görülmüştür. Jia vd. (2017) yaptıkları çalışmada çakışmasız OYA rotalama problemini çözmek için zaman penceresi metoduna dayalı robust şekilde çalışan geliştirilmiş A* algoritmasını önermişlerdir. Çalışmada çakışmasız en kısa rotalar hesaplanırken dönme faktörü de hesaba katılmıştır. Çalışmada çakışma problemi bekletme stratejisi ve yeniden rotalama stratejisi ile çözülmüştür. OYA'lar için hesaplanan rota uzunlukları zamanı ile zaman penceresi kullanılarak hesaplanan rota uzunlukları zamanı arasındaki fark dinamik olarak düzenlenebilmektedir. Önerilen algoritma küçük ve orta ölçekli sistemler için efektif rota hesaplaması yapabilmektedir.

Zhang vd. (2018) çakışmasız OYA rotalama problemini çözmek için zaman penceresi yöntemini ve A* algoritmasını kullanmışlardır. A* algoritması kullanılarak yapılan rota hesaplaması esnasında, çalışmada önerilen zaman penceresi yöntemi kullanılarak en az maliyetli çakışmasız rota seçilmeye çalışılır. Çalışmada kullanılan zaman penceresi yönteminde rota süreleri OYA'nın dönme faktörü de dikkate alınarak hesaplanır. Çalışmada meydana gelen çakışmaları önlemek için bekleme stratejisi kullanılmıştır. Çalışmada önerilen yöntem farklı boyutlara sahip ortamlarda da kullanılabilmektedir. Cui vd. (2019) yaptıkları çalışmada çakışmasız rota planlaması için klasik A* algoritmasını maliyet fonksiyonu açısından geliştirmişlerdir. Maliyet fonksiyonu hesabında bir OYA'nın düğüm üzerinde dönüş yaptığında hızında meydana gelen dalgalanmalar hesaba katılmıştır.

Çalışma sonunda önerilen algoritmanın maliyet fonksiyonu hesabını klasik A* algoritmasına göre daha hassas oranda yaptığı ortaya konulmuştur. Zhang vd. (2019) yaptıkları çalışmada OYA'ların rotalanmasında OYA dönüşlerinin sayısı ve yol planlama süresi dâhil olmak üzere hareket rotasının optimize edilmesi için geliştirilmiş A* algoritması önerilmektedir. A* algoritması tarafından elde edilen ilk rotaya dayalı olarak ikincil planlama için bir anahtar nokta seçim stratejisi kullanılır. Bu nedenle, yoldaki gereksiz dönme noktaları ve düğümleri etkin bir şekilde kaldırılabilir. Buna ek olarak, OYA'nın dönme noktasındaki dönüş yönü ve dönüş açısı bu şekilde belirlenebilir. Yapılan deneyler sonucunda geliştirilmiş A* algoritması, klasik A* algoritması ve karınca kolonisi algoritması gibi yöntemlere kıyasla daha kısa rotalar, daha az dönüş süresi ve daha kısa çalışma süresi ile daha verimli bir yol planlaması sağlayabilmektedir.

A* algoritması kullanarak çakışmasız rota problemini çözen bazı çalışmalarda, A* algoritmasının maliyet hesabındaki sezgisel fonksiyon geliştirilmiştir. Yuan vd. (2016) yaptıkları çalışmada OYA'ların çakışmasız rota planlaması için geliştirilmiş A* algoritmasını önermişlerdir. Sezgisel fonksiyon hesaplanırken manhattan uzaklığına ek olarak OYA'ların sayısı hedef noktasına olan uzaklıklarına bağlı olarak ağırlıklandırılıp eklenmiştir. Yöntemin sıralama etkinliği, orijinal A * algoritmasıyla karşılaştırılmıştır. Simülasyon sonuçları, yeni çakışma içermeyen yol planlama yönteminin, çoklu OYA sistemlerinin sıralama verimliliğini artırabileceğini ve trafik sıkışıklığını azaltabileceğini göstermektedir. Lin vd. (2017) yaptıkları çalışmada otonom robotların yol planlaması için geliştirilmiş A* algoritması önermişlerdir. Çalışmada önerilen A* algoritması maliyet hesabında sezgisel fonksiyon $h(n)$ değerine ek olarak, düğümün ebeveyn düğümünün de hedef düğüme olan sezgisel değeri hesaba katılmıştır. Ayrıca A* algoritmasındaki gerçek maliyet $g(n)$ ve sezgisel maliyet $h(n)$ değerlerin algoritmaya etkisi farklı ağırlıklarda hesaplanmıştır. Çalışmada Dijksta algoritması, A* algoritması ve geliştirilmiş A* algoritması yol uzunluğu, dolaşılan düğüm sayısı ve zaman bakımından karşılaştırılmıştır ve geliştirilmiş A* algoritması ile daha iyi sonuçlar elde edildiği gözlemlenmiştir.

Liu vd. (2017) yaptıkları çalışmada atölyelerdeki çoklu OYA'ların çizelgelenmesi için tek yönlü yönlendirilmiş çizge yöntemi ve çakışmasız rotalar oluşturmak için ise A * algoritması önermişlerdir. 20 OYA ile yapılan simülasyon deneyi önerilen yöntemin, OYA'ların çakışma problemini etkili bir şekilde çözebildiğini, istikrarlı ve gerçek zamanlı

olduğunu göstermektedir. Aslan ve Yazici (2020) yaptıkları çalışmada fabrikalardaki OTA'ların çakışmasız rota planlaması için A* algoritmasının geliştirilmiş versiyonunu önermişlerdir. Çalışmada önerilen algoritma beş OTA için test edilmiş ve meydana gelen çakışmalar tespit edilip çözülerek uygun rotalar oluşturulmuştur.

2.2.1. A* algoritması

Çizge üzerinde yol arama algoritmalarından biri olan A* algoritması, optimize çalışması sebebiyle bilgisayar bilimlerinde sıklıkla kullanılmaktadır (Lin vd., 2017). Hart vd. (1968) tarafından yayınlanan çalışmayla ortaya çıkan A* algoritması Dijkstra algoritmasının geliştirilmiş hali olarak görülmektedir. A* algoritması maliyet fonksiyonu hesabında gerçek maliyet ve sezgisel maliyet olmak üzere iki faktörü hesaba katar. Gerçek maliyet, OTA'nın aldığı yolun maliyeti (zaman, yol uzunluğu vb.) iken sezgisel maliyet hedef düğüme olan yaklaşık maliyeti ifade eder (Yuan vd., 2016). A* algoritması, Dijkstra algoritmasının gerçek maliyet hesabını, Best First Search (BFS) algoritmasının sezgisel maliyetini birleştirerek çalışan bir algoritma olarak düşünülebilir (Goyal vd., 2014). Sezgisel maliyet OTA'nın gideceği yönü belirlemede büyük etkiye sahiptir. Sezgisel maliyet ne kadar gerçek maliyete yaklaşırsa algoritmanın çalışma hızı o kadar artar. Sezgisel maliyetin gerçek maliyetten daha az olduğu durumlarda algoritma daha yavaş da olsa en uygun sonuca ulaşabilirken, daha hızlı olduğu durumlarda algoritma en uygun sonuca ulaşmada zorlanır (Yuan vd., 2016). A* algoritması ortamdaki herhangi bir nokta ile bitiş noktası arasındaki uzaklığı tahmin etmek için sezgisel maliyet fonksiyonu kullanır. Algoritma bitiş düğümüne ulaşana kadar her adımda en düşük maliyetli düğümü seçerek ilerler (Zhang vd., 2019).

A* algoritmasının sözde (pseudo) kodu Şekil 2.1'da verilmiştir (Zeng ve Church, 2009). Klasik A* algoritması, tek bir başlangıç ve bitiş noktası için en kısa yolu bulmayı amaçlamaktadır. Bu amaç doğrultusunda her iterasyonda en düşük maliyetli yolun seçilmesi hedeflenmektedir. Denklem 2.1'de verilen $f(n)$ maliyet fonksiyonu hesabı gerçek ve sezgisel maliyet fonksiyonu olmak üzere iki kısımdan oluşmaktadır. $g(n)$ ile gösterilen gerçek maliyet fonksiyonu başlangıç düğümünden bulunan düğüme olan maliyeti ifade etmektedir. Bu maliyet problemin tipine göre yol uzunluğu, yolun tamamlanma zamanı, yolda harcanan enerji vb. ölçüm parametrelerinden elde edilen değerler olabilir. $h(n)$ ile gösterilen sezgisel maliyet fonksiyonu ise, bulunan düğümden hedef düğümüne olan tahmini

maliyeti ifade eder (Hart vd., 1968). Sezgisel maliyet problem türüne göre değişebilmektedir. Örneğin yol uzunluğu hesaplanırken probleme göre Öklid, Manhattan, vb. uzaklıklar kullanılmaktadır.

$$f(n) = g(n) + h(n) \tag{2.1}$$

```
1.  A* Algoritması (başlangıç_düğümü, bitiş_düğümü)
2.  {
3.      KapalıListe = Ø, AçıkListe = Ø;
4.      başlangıç_düğümü.gerçek_maliyet = 0;
5.      başlangıç_düğümü.maliyet = başlangıç_düğümü.gerçek_maliyet + başlangıç_düğümü. sezgisel_maliyet;
6.      başlangıç_düğümü.ebeveyn = null;
7.      AçıkListe.Ekle(başlangıç_düğümü);
8.      do {
9.          v = AçıkListe.MinMaliyeliDüğümBul();
10.         AçıkListe.Kaldır(v);
11.         KapalıListe.Ekle(v);
12.         if (v == bitiş_düğümü )
13.         {
14.             return ;
15.         }
16.         else
17.         {
18.             foreach n_j in v.komşuluk_listesi
19.             {
20.                 if (n_j ∈ KapalıListe )
21.                 {
22.                     continue;
23.                 }
24.                 if ( (!(n_j ∈ AçıkListe )) &&( (v.gerçek_maliyet + maliyet(v, n_j) )< n_j.gerçek_maliyet))
25.                 {
26.                     n_j.gerçek_maliyet =v.gerçek_maliyet + maliyet(v, n_j);
27.                     n_j.maliyet = n_j.gerçek_maliyet + n_j.sezgisel_maliyet;
28.                     n_j.ebeveyn = v;
29.                     if (!(n_j ∈ AçıkListe))
30.                     {
31.                         AçıkListe.Ekle(n_j);
32.                     }
33.
34.                 }
35.
36.             }
37.         }
38.     } while (AçıkListe != Ø);
```

Şekil 2.1. A* Algoritması Sözde (Pseudo) Kodu

A* algoritması genellikle öncelikli kuyruk veri yapısı kullanılarak oluşturulan *AçıkListe* ve *KapalıListe* olmak üzere iki adet liste kullanır. AçıkListe gidilmesi muhtemel tüm düğüleri tutarken, KapalıListe değerlendirme fonksiyonuna göre seçilen düğümleri tutmaktadır. Algoritma başlangıç düğümünü *AçıkListe*'ye ekleyerek başlar. *AçıkListe*'deki her bir düğümün maliyet fonksiyonları hesaplanır. AçıkListe'deki en düşük maliyetli düğüm

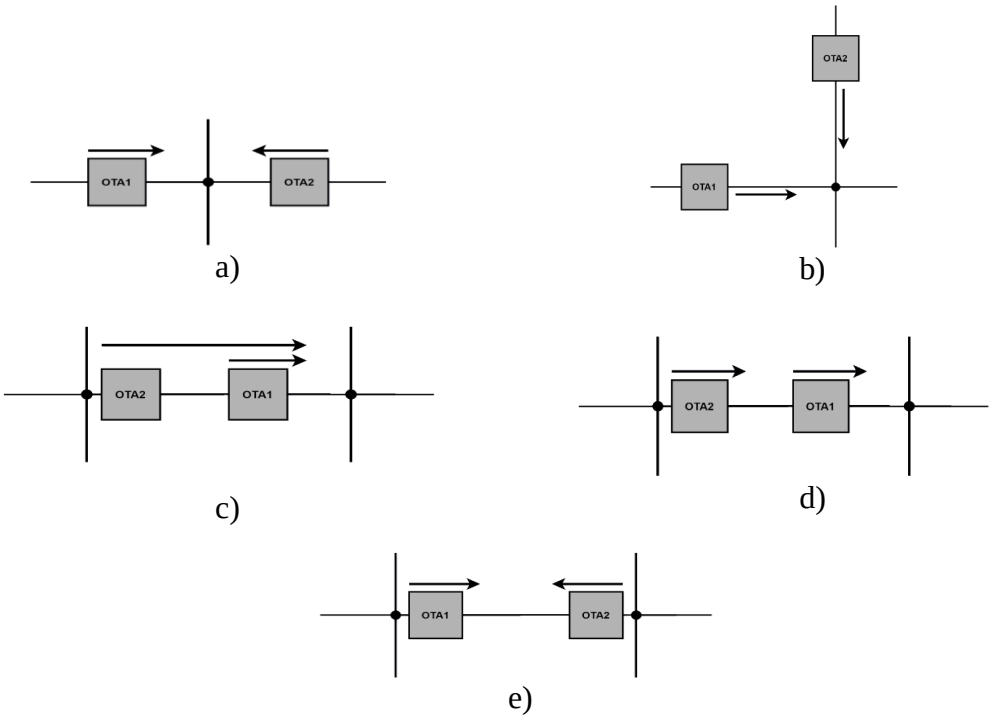
Açıkliste'den kaldırılarak *KapalıListe*'ye eklenir. *KapalıListe*'ye eklenen düğüm hedef düğüm ise algoritma çalışmayı bitirir, değilse bu düğümün komşu düğümleri AçıkListe'ye eklenir. Bu döngü hedef düğüme ulaşılıncaya kadar devam eder.

3. MATERYAL VE YÖNTEM

Bu bölümde ilk olarak tez çalışmasında ele alınan tüm çakışma tipleri ayrıntılı olarak tanımlanmıştır. Daha sonra tez kapsamında önerilen algoritma detaylıca açıklanmıştır. Son olarak test ortamı tanıtılarak önerilen algoritmanın örnekler ile nasıl çalıştığı gösterilmiştir.

3.1. Çakışma Tipleri

Ortamdaki n adet düğüm için her bir düğüm d , toplam düğüm kümesi ise $D=\{d_1,d_2,....d_n\}$ olsun. Ortamda hareket eden k adet OTA $R_{1\to k}$, her bir düğüme ait koordinatlar $(X_{1\to n},Y_{1\to n})$, R_k OTA'sının bulunduğu düğüme ait koordinatlar $R_k(X_n,Y_n)$, R_k OTA'nın hızı $R_k(s)$ olarak sembolize edilmektedir. R_k OTA'sının, d_n düğümüne geliş zamanı $T_n(R_k^{tg})$, d_n düğümünden çıkış zamanı $T_n(R_k^{tc})$ ile gösterilmektedir.

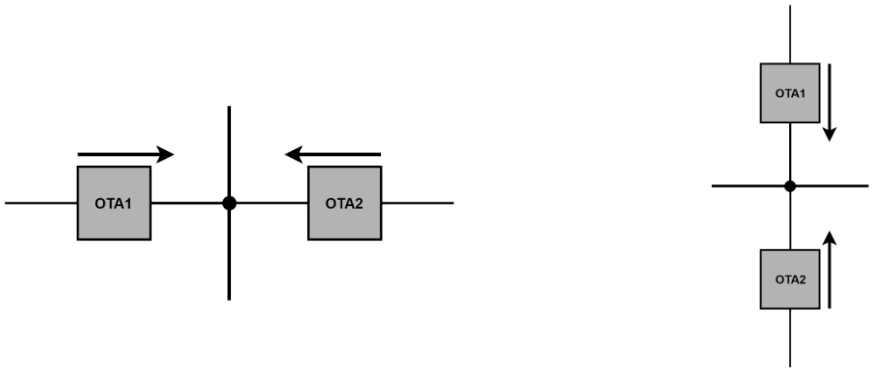


Şekil 3.1. Tüm Çakışma Tipleri: a) Aynı Doğrultuda Düğüm Üzerinde Karşılıklı Çakışma, b) Farklı Doğrultuda Düğüm Üzerinde Karşılıklı Çakışma, c) Hız Farkından Kaynaklanan Arka Arkaya Düğüm Çakışması, d) Hız Farkından Kaynaklanan Arka Arkaya Kenar Çakışması, e) Kenar Üzerinde Karşılıklı Çakışma

Tez çalışmasında çakışma tipleri genel olarak iki başlık altında incelenmektedir. Düğüm ve kenar çakışması olarak adlandırılan bu iki çakışma tipine ait beş çakışma tipi Şekil 3.1'de gösterilmektedir.

3.1.1. Aynı doğrultuda düğüm üzerinde karşılıklı çakışma

Aynı doğrultu üzerinde hareket eden farklı düğümlerden gelen R_1 ve R_2 OTA'ları aynı zaman penceresinde aynı düğümü kullanmak isterlerse Şekil 3.2'de gösterilen çakışma tipi meydana gelir.



Şekil 3.2. Aynı Doğrultuda Düğüm Üzerinde Karşılıklı Çakışma

n düğümünden hareket eden R_1 ile m düğümünden hareket eden R_2 olarak adlandırılan iki OTA olsun. Bu OTA'lar (3.1), (3.2), (3.3) ve (3.6) denklemlerini sağlarsa aynı doğrultuda düğüm üzerinde karşılıklı çakışma meydana gelir. Denklem (3.1) ve (3.2) OTA'ların farklı düğümlerden gelip aynı düğüme doğru hareket ettiklerini, denklem (3.3) OTA'ların aynı doğrultuda hareket ettiklerini ifade etmektedir.

$$R_1(X_{n+1}, Y_{n+1}) = R_2(X_{m+1}, Y_{m+1}) \tag{3.1}$$

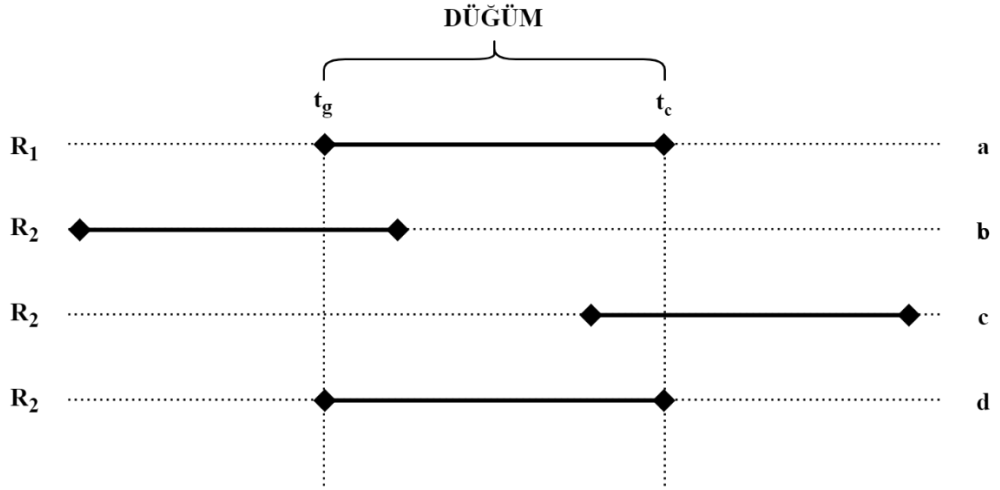
$$R_1(X_n, Y_n) \neq R_2(X_m, Y_m) \tag{3.2}$$

$$(R_1(X_n) = R_2(X_m)) \,\|\, (R_1(Y_n) = R_2(Y_m)) \tag{3.3}$$

$$\alpha = \left[T_{n+1}(R_1^{tg}) \geq T_{m+1}(R_2^{tg}) \,\&\&\, T_{n+1}(R_1^{tg}) \leq T_{m+1}(R_2^{tc}) \right] \tag{3.4}$$

$$\beta = \left[T_{n+1}(R_1^{tc}) \geq T_{m+1}(R_2^{tg}) \,\&\&\, T_{n+1}(R_1^{tc}) \leq T_{m+1}(R_2^{tc}) \right] \tag{3.5}$$

$$\left(\alpha \,\|\, \beta\right) \tag{3.6}$$



Şekil 3.3. Düğüm Çakışma Tipleri İçin Zaman Penceresi Gösterimi

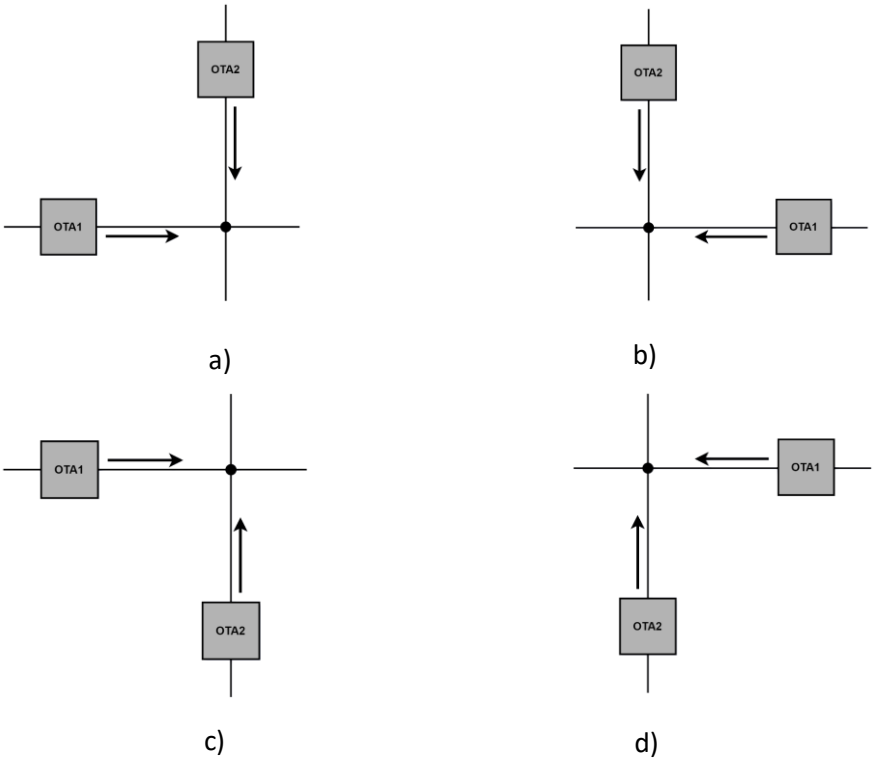
Denklem (3.4) zaman penceresi gösterimi verilen Şekil 3.3'ü ifade etmektedir. Şekil 3.3'teki a durumu R_1 OTA'sının herhangi bir düğüme giriş (t_g) ve çıkış (t_c) zamanını göstermektedir. Şekil 3.3'teki b durumunda R_2 OTA'sı düğümdeyken R_1 OTA'sı düğüme geldiğinden ötürü çakışma meydana gelmektedir (Denklem 3.4). Şekil 3.3'teki c durumunda R_1 OTA'sı düğümdeyken R_2 OTA'sı düğüme geldiğinden ötürü çakışma meydana gelmektedir (Denklem 3.5). Şekil 3.3'teki d durumunda ise R_1 ve R_2 OTA'larının düğüme giriş-çıkış zaman pencereleri eşit olduğundan dolayı çakışma meydana gelmektedir (Denklem 3.6). Önerilen algoritma tüm OTA'lar için düğüm bekleme zamanını ($T\_dügüm\_bekleme$) sabit ve eşit olarak ele aldığından dolayı düğüm çakışması için Şekil 3.3'te verilen durumlar dışında başka bir durumun meydana gelmesi beklenemez.

3.1.2. Farklı doğrultuda düğüm üzerinde karşılıklı çakışma

Düğüm çakışma tiplerinden olan farklı doğrultuda düğüm üzerinde karşılıklı çakışma durumu, farklı doğrultularda hareket eden R_1 ve R_2 OTA'larının aynı zaman diliminde aynı düğümü kullanmak istemesi şeklinde tanımlanmaktadır.

Bu OTA'lar (3.1), (3.2), (3.6) ve (3.7) denklemlerini sağlarsa farklı doğrultuda düğüm üzerinde karşılıklı çakışma meydana gelir. Şekil 3.4'te bu çakışma tipine ait meydana

gelebilecek tüm durumlar verilmiştir. Denklem (3.7) OTA'ların farklı doğrultu üzerinde hareket ettiğini ifade etmektedir.



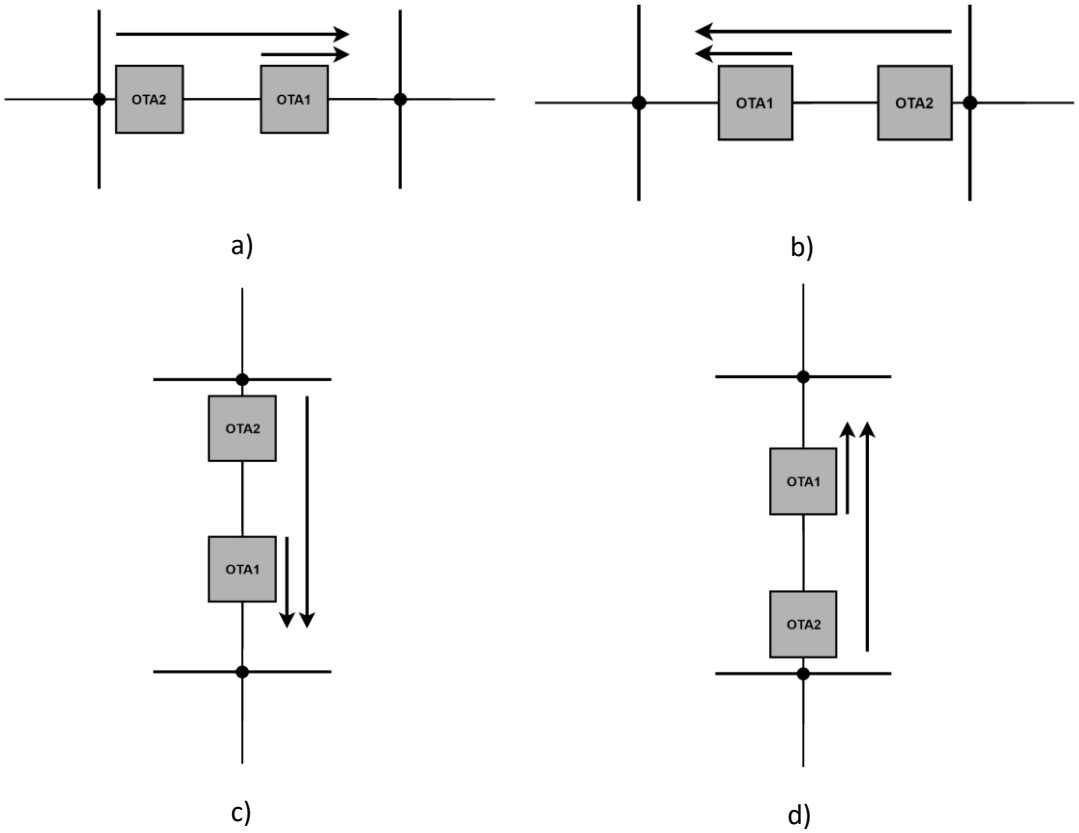
Şekil 3.4. Farklı Doğrultuda Düğüm Üzerinde Karşılıklı Çakışma

$$(R_1(X_n) \neq R_2(X_m)) \,\&\&\, (R_1(Y_n) \neq R_2(Y_m)) \tag{3.7}$$

3.1.3. Hız farkından kaynaklanan arka arkaya düğüm çakışması

Düğüm çakışma tiplerinden olan hız farkından kaynaklanan arka arkaya düğüm çakışması durumu, aynı doğrultuda hareket eden ve aynı düğümden gelen OTA'ların aynı zaman diliminde aynı düğümü kullanmak istemesi şeklinde tanımlanmaktadır. Bu OTA'lar (3.1), (3.6), (3.8) ve (3.9) denklemlerini sağlarsa hız farkından kaynaklanan arka arkaya düğüm çakışması meydana gelir. Şekil 3.5'te bu çakışma tipine ait meydana gelebilecek tüm durumlar verilmiştir.

Denklem (3.8) R_1 ve R_2 OTA'larının çakışmadan önce en son aynı düğümden hareket ettiklerini ifade etmektedir. Denklem (3.9) ise arkadan gelen R_2 OTA'sının hızının öndeki R_1 OTA'nın hızından daha fazla olması gerektiğini ifade etmektedir.



Şekil 3.5. Hız Farkından Kaynaklanan Arka Arkaya Düğüm Çakışması

$$R_1(X_n, Y_n) = R_2(X_m, Y_m) \tag{3.8}$$

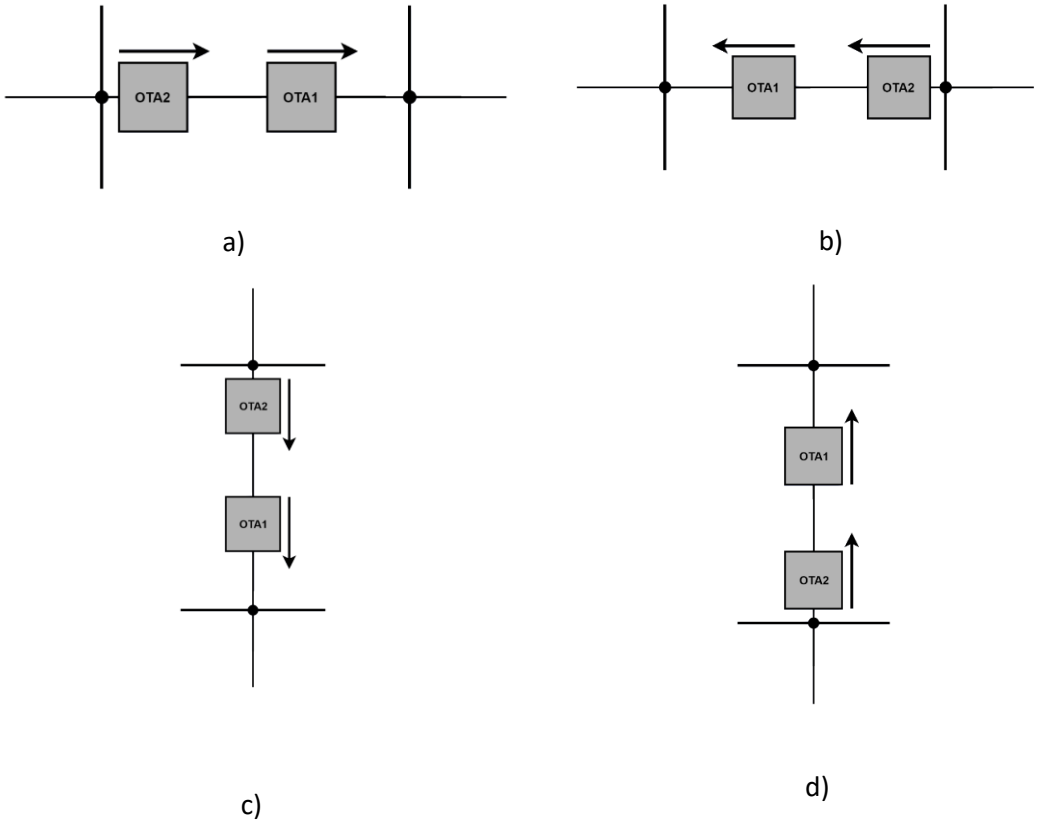
$$R_1(s) < R_2(s) \tag{3.9}$$

3.1.4. Hız farkından kaynaklanan arka arkaya kenar çakışması

Kenar çakışma tiplerinden olan hız farkından kaynaklanan arka arkaya kenar çakışması durumu, aynı doğrultuda hareket eden ve aynı düğümden gelen OTA'ların aynı zaman diliminde aynı kenarı kullanmak istemesi şeklinde tanımlanmaktadır. Şekil 3.6'da bu çakışma tipine ait meydana gelebilecek tüm durumlar verilmiştir. Bu çakışma tipinde iki farklı durum mevcuttur. İlk durum yavaş olan OTA'nın *ÇakışmaKontrolKenarListesi*'ne

daha önce girmesi durumudur. Bu durumda hızlı olan OTA'nın çakışma olan kenara girmesi durumunda önde olan yavaş OTA ile çakışma olmaması için hızlı OTA'nın rotası değiştirilir veya bekletilir. Bu durum (3.1), (3.8), (3.9), (3.14) ve (3.17) denklemlerinin sağlanmasıyla ortaya çıkmaktadır. İkinci durum hızlı olan OTA'nın *ÇakışmaKontrolKenarListesi*'ne daha önce girmesi durumudur. Bu durumda işlem yapılacak olan yavaş OTA *ÇakışmaKontrolKenarListesi*'ne daha sonra girmesine rağmen kenara daha önce girmektedir. Böyle bir durumda yavaş olan OTA üzerinde işlem yapıldığından dolayı OTA'nın rotası zorunlu olarak değiştirilmektedir. Bu durum (3.1), (3.8), (3.9), (3.14) ve (3.19) denklemlerinin sağlanmasıyla ortaya çıkmaktadır.

Denklem (3.14) zaman penceresi gösterimi verilen Şekil 3.7'yi ifade etmektedir. Şekil 3.7'deki a durumu R_1 OTA'sının herhangi bir kenara giriş (t_g) ve çıkış (t_c) zamanını göstermektedir. Şekil 3.7'deki b durumunda R_1 OTA'sı kenara R_1 OTA'sından erken girip geç çıkması durumunda çakışma meydana gelmektedir (Denklem 3.10 ve 3.11).



Şekil 3.6. Hız Farkından Kaynaklanan Arka Arkaya Kenar Çakışması

$$\delta = \left[T_n(R_1^{tc}) \geq T_m(R_2^{tc}) \,\&\&\, T_n(R_1^{tc}) \leq T_{m+1}(R_2^{tg}) \right] \tag{3.10}$$

$$\varepsilon = \left[T_{n+1}(R_1^{tg}) \geq T_m(R_2^{tc}) \,\&\&\, T_{n+1}(R_1^{tg}) \leq T_{m+1}(R_2^{tg}) \right] \tag{3.11}$$

$$\gamma = \left[T_m(R_2^{tc}) \geq T_n(R_1^{tc}) \,\&\&\, T_m(R_2^{tc}) \leq T_{n+1}(R_1^{tg}) \right] \tag{3.12}$$

$$\theta = \left[T_{m+1}(R_2^{tg}) \geq T_n(R_1^{tc}) \,\&\&\, T_{m+1}(R_2^{tg}) \leq T_{n+1}(R_1^{tg}) \right] \tag{3.13}$$

$$\left[(\delta \,\|\, \varepsilon) \| (\gamma \,\|\, \theta) \right] \tag{3.14}$$

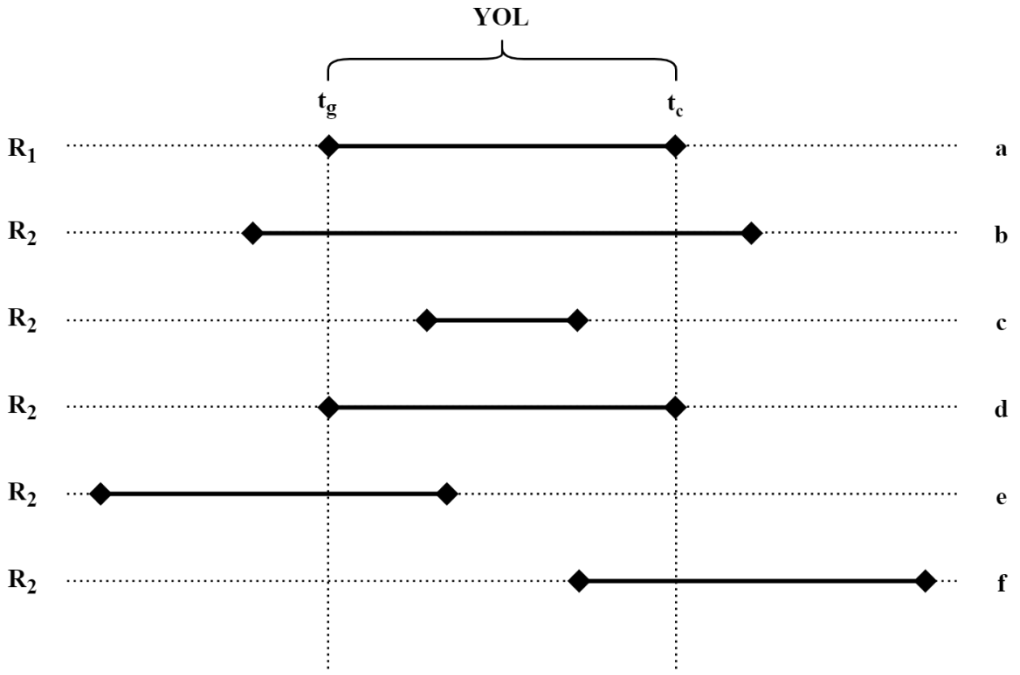
$$\Psi = \left[\left(T_n(R_1^{tc}) > T_m(R_2^{tc}) \right) \&\& \left(T_{n+1}(R_1^{tg}) < T_{m+1}(R_2^{tg}) \right) \right] \tag{3.15}$$

$$\Upsilon = \left[\mathit{AçikListe}\left(R(s)\right) > \mathit{ÇakismaKontrolYolListesi}\left(R(s)\right) \right] \tag{3.16}$$

$$\left(\Psi \,\&\&\, \Upsilon \right) \tag{3.17}$$

$$\Theta = \left[\left(T_n(R_1^{tc}) < T_m(R_2^{tc}) \right) \&\& \left(T_{n+1}(R_1^{tg}) > T_{m+1}(R_2^{tg}) \right) \right] \tag{3.18}$$

$$\left(\Theta \,\&\&\, \overline{\Upsilon} \right) \tag{3.19}$$



Şekil 3.7. Kenar Çakışma Tipleri İçin Zaman Penceresi Gösterimi

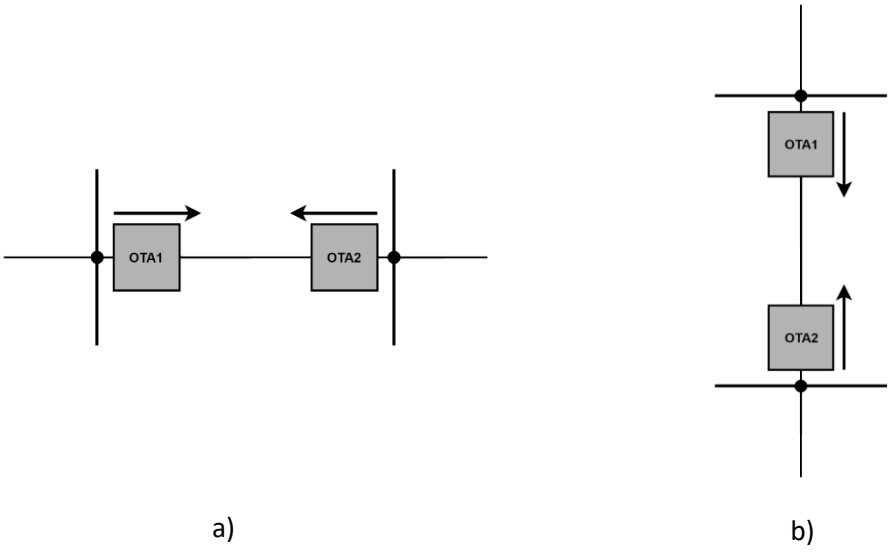
Şekil 3.7'deki c durumunda R_2 OTA'sı kenara R_1 OTA'sından geç girip erken çıkması durumunda çakışma meydana gelmektedir (Denklem 3.12 ve 3.13). Şekil 3.7'deki d

durumunda ise R_1 ve R_2 OTA'ları aynı anda kenara giriş çıkış yapmaya çalıştıklarından dolayı çakışma meydana gelmektedir (Denklem 3.14).

Tez kapsamında önerilen algoritma çakışma olmaması koşuluyla aynı kenarda birden fazla OTA'nın bulunmasına optimumluğu daha fazla sağlaması açısından izin vermektedir. Şekil 3.7'deki e durumunda R_2 OTA'sı kenarı terk etmeden R_1 OTA'sı yola gelmektedir. Fakat R_1 OTA'sı kenar boyunca R_2 OTA'sına kavuşamamaktadır. Şekil 3.7'deki f durumunda ise R_1 OTA'sı kenarı terk etmeden R_2 OTA'sı kenara gelmektedir. Fakat R_2 OTA'sı kenar boyunca R_1 OTA'sına kavuşamamaktadır. Bu sebeplerden dolayı her iki durumda da çakışma meydana gelmemektedir.

3.1.5. Kenar üzerinde karşılıklı çakışma

Kenar çakışma tiplerinden olan kenar üzerinde karşılıklı çakışma durumu, aynı doğrultuda hareket eden ve farklı düğümden gelen OTA'ların aynı zaman diliminde aynı kenarı kullanmak istemesi şeklinde tanımlanmaktadır. Bu OTA'lar (3.14), (3.20) ve (3.21) denklemlerini sağlarsa kenar üzerinde karşılıklı çakışma meydana gelir. Şekil 3.8'de bu çakışma tipine ait meydana gelebilecek tüm durumlar verilmiştir.



Şekil 3.8. Kenar Üzerinde Karşılıklı Çakışma

$$R_1(X_{n+1}, Y_{n+1}) = R_2(X_m, Y_m) \quad (3.20)$$

$$R_1(X_n, Y_n) = R_2(X_{m+1}, Y_{m+1}) \quad (3.21)$$

Denklem (3.20) R_1 OTA'sının varış düğümü ile R_2 OTA'sının çıkış düğümünün aynı düğüm olduğunu, denklem (3.21) ise R_1 OTA'sının çıkış düğümü ile R_2 OTA'sının varış düğümünün aynı düğüm olduğunu ifade etmektedir.

3.2. Yöntem

Fabrika ortamında yüklerin belirli bir noktadan başka bir noktaya taşınması işlemi gelişen teknoloji sayesinde OTA'lar tarafından yapılmaya başlanmıştır. Aynı anda birden fazla OTA'nın çakışma olmadan mümkün olduğunca optimum şekilde hareket ettirilmesi için literatürde farklı yöntemler mevcuttur. Bu bağlamda tez çalışması kapsamında belirtilen bu problemi çözmek için A* temelli yeni bir algoritma önerilmiştir. Önerilen algoritma klasik A* algoritmasını geliştirerek aynı anda birden fazla heterojen OTA için çakışmasız uygun rotaları hesaplayabilmektedir.

Önerilen algoritmanın sözde (pseudo) kodu Şekil 3.9'da verilmiştir. Algoritmanın 2 ile 11. satırları arasında kullanılan veri yapıları Çizelge 3.1'de tanıtılmıştır. Tez kapsamında kullanılan ortam düğüm ve kenar olarak adlandırılan iki farklı veri yapısı kullanılarak oluşturulmuştur. Önerilen algoritma bu iki veri yapısı üzerinden çalışmaktadır. Algoritmadaki düğüm ve kenar veri yapıları için oluşturulan özellikler Çizelge 3.1'de gösterilmiştir. Algoritma tüm OTA'lar için ortak birer *AçıkListe*, *KapalıListe* ve *ÇakışmaKontrolKenarListesi* kullanmaktadır. *AçıkListe* bulunulan düğümden gidilebilecek tüm komşu düğümleri tutan bir listedir. *KapalıListe* gidilmesine karar verilmiş en düşük maliyetli düğümleri tutan bir listedir. *ÇakışmaKontrolKenarListesi* ise meydana gelebilecek kenar çakışmalarının kontrolü için oluşturulmuştur. Bu liste gidilmesi muhtemel ve gidilecek tüm kenarları tutmaktadır. Algoritmada kullanılan bu üç liste de öncelikli kuyruk veri yapısı kullanılarak tasarlanmıştır.

Çizelge 3.1. Düğüm ve Kenar Veri Yapıları İçin Oluşturulan Özellikler

Düğüm	**Kenar**
g :gerçek maliyet	*uzunluk* : komşu iki düğüm arasındaki mesafe
h :sezgisel maliyet	$baglanti\_dügümü1$: kenarı oluşturan düğümlerden biri
t_g :düğüme geliş zamanı	$baglanti\_dügümü2$:kenarı oluşturan düğümlerden biri
t_c :düğümden çıkış zamanı	t_g :kenara giriş zamanı
ebeveyn :ata düğüm	t_c :kenardan çıkış zamanı
$bekleme\_zamani$:çakışmadan kaynaklı beklenen zaman	$kenarın\_gidilme\_durumu$: OTA tarafından kenarın kullanılıp kullanılmama durumu
$komsu\_kenarlar$: düğümden gidilebilecek komşu kenarlar	

Önerilen algoritma ortama göre sayı sınırlaması olmaksızın çoklu heterojen OTA'lar için çalışmaktadır. R_1 ve R_2 olarak adlandırılan iki OTA için algoritmanın çalışma prensibi bu bölümün geri kalanında açıklanmıştır. $d_{1\rightarrow 36}$ düğümlerine sahip ortamda $d_{baslangic}$ düğümünden d_{bitis} düğümüne giden d_n düğümündeki bir k OTA'sı için maliyet fonksiyonu Denklem (3.22)'de gösterilmektedir. Denklem (3.22)'deki $g(d_n)$, başlangıç düğümünden bulunulan düğüme olan gerçek maliyeti, $h(d_n)$ bulunulan düğümden bitiş düğüme olan sezgisel maliyeti, $f(d_n)$ ise bitiş düğümüne ulaşmak için hesaplanan toplam maliyeti ifade etmektedir. Denklem (3.23)'te gerçek maliyetin nasıl hesaplandığı gösterilmektedir. Buradaki d_n^x ve d_n^y ortamdaki x ve y konumlarını ifade etmektedir. Denklem (3.24)'te sezgisel maliyetin nasıl hesaplandığı gösterilmektedir ve sezgisel maliyet hesabında iki düğüm arasındaki uzaklık ölçüm yöntemi olarak Manhattan uzaklık kullanılmıştır. İki noktanın kartezyen koordinatları arasındaki mutlak değer farkı olarak hesaplanan Manhattan uzaklığı, ızgara (grid) tabanlı ortamlarda kullanılabildiğinden dolayı algoritmada Manhattan uzaklığı seçilmiştir.

$$f(d_n) = g(d_n) + h(d_n) \tag{3.22}$$

$$g(d_n) = (|\, d_n^x - d_{baslangic}^x \,| + |\, d_n^y - d_{baslangic}^y \,|) / R_k(s) \tag{3.23}$$

$$h(d_n) = (|\, d_n^x - d_{bitis}^x \,| + |\, d_n^y - d_{bitis}^y \,|) / R_{hizli}(s) \qquad (3.24)$$

```
1.  Önerilen A* Algoritması (başlangıç_düğümleri, bitiş_düğümleri, T_düğüm_bekleme)
2.  {
3.      KapalıListe = Ø, ÇakışmaKontrolKenarListesi = Ø, AçıkListe = Ø;
4.      foreach başlangıç_düğümü in başlangıç_düğümleri
5.      {
6.          başlangıç_düğümü.gerçek_maliyet = 0;
7.          başlangıç_düğümü.maliyet = başlangıç_düğümü.gerçek_maliyet + başlangıç_düğümü. sezgisel_maliyet;
8.          başlangıç_düğümü.ebeveyn = null;
9.          başlangıç_düğümü.Tg = 0;
10.         başlangıç_düğümü.Tc = T_düğüm_bekleme;
11.         AçıkListe.Ekle(başlangıç_düğümü);
12.     }
13.     do {
14.         v = AçıkListe.MinMaliyeliDüğümBul();
15.         DüğümÇakışmaKontrol(KapalıListe, AçıkListe,v);
16.         KenarÇakışmaKontrol(ÇakışmaKontrolKenarListesi, AçıkListe,v);
17.         v = AçıkListe.MinMaliyeliDüğümBul();
18.         foreach ej in ÇakışmaKontrolKenarListesi
19.         {
20.             if (v.ebeveyn == ej.bağlantı_düğümü1 && v == ej.bağlantı_düğümü2)
21.             {
22.                 ej.gidilen_kenar = true;
23.             }
24.         }
25.         AçıkListe.Kaldır(v);
26.         foreach ej in v.komşukenarlar
27.         {
28.             if ((ej. bağlantı_düğümü1.gercek_maliyet == null) || (v.gercek_maliyet + ej.uzunluk / ej. bağlantı_düğümü1.hız) < ej.
                    bağlantı_düğümü1.gercek_maliyet)
29.             {
30.                 ej. bağlantı_düğümü1.gercek_maliyet =v.gercek_maliyet + ej.uzunluk / ej. bağlantı_düğümü1.hız;
31.                 ej. bağlantı_düğümü1.maliyet= ej. bağlantı_düğümü1.gercek_maliyet+ ej. bağlantı_düğümü1.sezgisel_maliyet;
32.                 ej. bağlantı_düğümü1.ebeveyn = v;
33.                 ej. bağlantı_düğümü1.Tg = ej. bağlantı_düğümü1.gercek_maliyet;
34.                 ej. bağlantı_düğümü1.Tc =ej. bağlantı_düğümü1.gercek_naliyet + T_ düğüm_bekleme;
35.             }
36.             if (!(ej.bağlantı_düğümü1 ∈ AçıkListe))
37.             {
38.                 AçıkListe.Ekle(ej.bağlantı_düğümü1);
39.                 kenar.bağlantı_düğümü1= ej.bağlantı_düğümü1.ebeveyn;
40.                 kenar.bağlantı_düğümü2 = ej.bağlantı_düğümü1;
41.                 ÇakışmaKontrolKenarListesi.Ekle(kenar);
42.             }
43.         }
44.         KapalıListe.Ekle(v);
45.         foreach nj in bitiş_düğümleri
46.         {
47.             if (v == nj)
48.             {
49.                 foreach mj in AçıkListe
50.                 {
51.                     if (mj.OTA_ismi == v. OTA_ismi)
52.                     {
53.                         AçıkListe.Kaldır(mj);
54.                     }
55.                 }
56.             }
57.         }
58.     } while (AçıkListe != Ø);
59. }
```

Şekil 3.9. Önerilen A* Algoritması Sözde (Pseudo) Kodu

Denklem (3.24)'de $R_{hizli}(s)$ ortamdaki OTA'lardan en hızlı OTA'nın hızını ifade etmektedir. Bir OTA'nın kendi hızı Denklem (3.23) ve (3.24) 'de görüleceği üzere toplam maliyeti etkilemektedir. Heterojen OTA'lar arasındaki hız farklı OTA'ların toplam maliyetleri arasında büyük farklar meydana getirebilmektedir. Önerilen algoritma birden fazla OTA için aynı anda rota bulmaya çalışmasından dolayı *AçıkListe* ve *KapalıListe'de* farklı OTA'lara ait düğümler bulunabilmektedir. Bu durumda diğer OTA'lara oranla hızı aşırı olan OTA çoğunlukla en düşük toplam maliyete sahip olur. Dolayısıyla algoritma her adımda bu OTA'ya ait düğümleri seçer. Bu durumda bu OTA sürekli olarak öncelik elde edeceğinden dolayı tüm robotlar için optimum sonuca ulaşmak zorlaşır. Sonuç olarak her adımda seçilecek en düşük maliyetli düğümün OTA'lara daha dengeli dağılmasını sağlamak için tüm OTA'lardaki sezgisel maliyet hesabında en hızlı robotun hızı hesaba katılmaktadır.

Algoritma OTA'ların başlangıç ve bitiş düğümlerini bir liste halinde parametre olarak alır. Her OTA'nın başlangıç düğüme ait özelliklerin ilk değer atamaları yapılır. Burada başlangıç düğümünün gerçek maliyeti sıfır (0) olarak atanır. Sezgisel maliyet ise Denklem (3.24)'deki formül kullanılarak hesaplanır. Başlangıç düğümünün ebeveyn düğümü bulunmadığından bu değer *null* olarak atanır. Başlangıç düğümüne geliş zamanı (t_g) sıfır (0) olarak atanır. Başlangıç düğümünden çıkış zamanı (t_c) ise düğüme giriş zamanına düğümde bekleme zamanının ($T\_dügüm\_bekleme$) eklenmesiyle oluşturulmaktadır. Tüm bu başlangıç düğümleri *AçıkListe*'ye eklenir.

AçıkListe'deki düğümler toplam maliyetlerine göre küçükten büyüğe doğru sıralanır. İlk olarak en düşük maliyetli düğüm gidilecek aday düğüm olarak seçilir. Bu aday düğüm, düğüm çakışma kontrolü için Şekil 3.9'un 15. satırında yer alan *DüğümÇakışmaKontrol* fonksiyonu kullanılarak incelenir. Bu fonksiyon ayrıntılı olarak bölüm 3.2.1'de açıklanmıştır. Düğüm çakışması olması durumunda çakışma tipine göre aday düğüm *AçıkListe*'den kaldırılır veya aday düğümün maliyeti güncellenir. Daha sonra bu aday düğüm kenar çakışma kontrolü için Şekil 3.9'un 16. satırında yer alan *KenarÇakışmaKontrol* fonksiyonu kullanılarak çakışma durumu incelenir. Bu fonksiyon ayrıntılı olarak bölüm 3.2.2'de açıklanmıştır. Kenar çakışması olması durumunda bu fonksiyon *AçıkListe'yi* ve *ÇakışmaKontrolKenarListe'*sini değiştirmektedir. Bu sebeple değişen *AçıkListe*'deki düğümler tekrar toplam maliyetlerine göre küçükten büyüğe sıralanır. En küçük maliyetli düğüm gidilecek aday düğüm olarak *AçıkListe'*den kaldırılarak *KapalıListe'*ye eklenmek

üzere seçilir. Seçilen bu düğüm ile ebeveyn düğümü arasındaki kenar *ÇakışmaKontrolKenarListe'*si içerisinde *yolun_gidilme_durumu* özelliği *true* olarak işaretlenir. Buradaki amaç kenar çakışması kontrolü yapılırken kenarın o an için bir OTA tarafından işgal edilip edilmediğini öğrenmektir.

KapalıListe'ye eklenmek üzere seçilen bu düğümün belirli şartları sağlayan komşu düğümleri *AçıkListe*'ye eklenir. Eklenecek olan bu komşu düğümler *AçıkListe*'deki düğümler ile karşılaştırılır. Karşılaştırma yapılacak düğümlerin aynı OTA'ya ait olması gerekmektedir. Eklenecek olan bu komşu düğümün aynısı *AçıkListe*'de varsa, bu düğümle var olan düğüm toplam maliyet açısından karşılaştırılır ve eklenecek düğümün maliyetinin daha küçük olması durumunda var olan düğüm *AçıkListe*'den kaldırılarak eklenecek komşu düğüm özellikleri ayarlanarak *AçıkListe*'ye yeniden eklenir. Aksi durumda *AçıkListe* değiştirilmez. Komşu düğümleri *AçıkListe*'ye eklenen aday düğüm *KapalıListe*'ye eklenir. *KapalıListe*'ye eklenen düğümün bitiş düğümü olup olmadığı kontrol edilir. Eğer bitiş düğümü ise bu düğümün ait olduğu OTA tespit edilir ve bu OTA'ya ait *AçıkListe*'de bulunan tüm düğümler *AçıkListe*'den kaldırılır. Algoritma *AçıkListe*'de eleman kalmayıncaya kadar çalışmaya devam eder.

3.2.1. Düğüm çakışma kontrolü

İki tip çakışma türünden biri olan düğüm çakışması için Şekil 3.10'da sözde kodu verilen fonksiyon çözüm üretebilmektedir. *KapalıListe* gidilebilecek düğümleri tuttuğundan dolayı düğüm çakışma kontrolleri *KapalıListe* üzerinden yapılmaktadır. *AçıkListe*'den elde edilen aday düğüm, *KapalıListe*'deki tüm düğümler ile düğüm çakışması kontrolü için karşılaştırılır. Farklı doğrultuda düğüm üzerinde karşılıklı çakışma, aynı doğrultuda düğüm üzerinde karşılıklı çakışma ve hız farkından kaynaklanan arka arkaya düğüm çakışması olmak üzere üç tip düğüm çakışma durumu vardır.

3.2.1.1. Aynı doğrultuda düğüm üzerinde karşılıklı çakışma

Bölüm 3.1.1'de ayrıntılı olarak açıklanan aynı doğrultuda düğüm üzerinde karşılıklı çakışma tipi Şekil 3.10'da verilen sözde kodun 7. satırında algoritmik olarak ifade edilmiştir. Burada 7. satırdaki ifade karşılıklı çakışmanın genel koşulunu ifade ederken, 9. satırda

verilen ifade aynı doğrultuda düğüm üzerinde karşılıklı çakışma koşulunu ifade etmektedir. Şekil 3.10'daki sözde kodun 12. ve 21. satırları arasında *AçıkListe* içerisinde çakışma kontrolü yapılan düğüm (v), *KapalıListe* içerisinde çakışma meydana gelen düğüm (d_k) karşılaştırılır. Burada d_k

```
1.  DüğümÇakışmaKontrol(KapalıListe, AçıkListe,v)
2.  {
3.      foreach n_j in KapalıListe
4.      {
5.          bekleme_zamanı = n_j. T_c – v. T_g + T_düğüm_bekleme;
6.
7.          if (v.ebeveyn!=n_j .ebeveyn && v ==  n_j && ((v. T_g >= n_j T_g && v. T_g <= n_j T_c) || (v. T_c >= n_j T_g && v. T_c <= n_j T_c)))
8.          {
9.              if ( (v.x_koordinat== n_j. x_koordinat ) || (v.y_koordinat== n_j. y_koordinat))
10.             {
11.                 kontrol=0;
12.                 foreach d_k in KapalıListe
13.                     if (v.ebeveyn== d_k && n_j== d_k)
14.                     {
15.                         AçıkListe.Kaldır(v);
16.                         kontrol=1;
17.                         break;
18.                     }
19.                 if  (k<= KapalıListe.uzunluk)
20.                     AçıkListe.Kaldır(v);
21.                 if  (kontrol==0)
22.                 {
23.                     v.gercek_maliyet = v.gercek_maliyet + bekleme_zamanı;
24.                     v. T_g = v. T_g + bekleme_zamanı;
25.                     v. T_c = v. T_c + bekleme_zamanı;
26.                 }
27.
28.             }
29.             else
30.             {
31.                 v.gercek_maliyet = v.gercek_maliyet + bekleme_zamanı;
32.                 v. T_g = v. T_g + bekleme_zamanı;
33.                 v. T_c = v. T_c + bekleme_zamanı;
34.             }
35.         }
36.         if (v.ebeveyn==n_j .ebeveyn && v ==  n_j && ((v. T_g >= n_j T_g && v. T_g <= n_j T_c) || (v. T_c >= n_j T_g && v. T_c <= n_j T_c)))
37.         {
38.             if (v.hız>n_j.hız)
39.             {
40.                 v.gercek_maliyet = v.gercek_maliyet + bekleme_zamanı;
41.                 v. T_g = v. T_g + bekleme_zamanı;
42.                 v. T_c = v. T_c + bekleme_zamanı;
43.             }
44.             else
45.             {
46.                 AçıkListe.Kaldır(v);
47.             }
48.         }
49.     }
50. }
```

Şekil 3.10. Düğüm Çakışma Kontrolü Fonksiyonu Sözde (Pseudo) Kodu

düğümünün bir sonraki düğümü, v düğümüne eşitse v düğümü *AçıkListe'*den kaldırılır. Bu durumda işlem yapılan OTA'nın çakışmanın olduğu düğüme gitme ihtimali ortadan

kaldırılmış olur. Bunun dışında kalan durumlarda v düğümünün maliyeti artırılır (23-25. satır arası). Bu durumda maliyeti güncellenen düğüm hala *AçıkListe*'deki en düşük maliyetli düğüm ise işlem yapılan OTA bekletilmiş olur, değilse rota değiştirmiş olur.

3.2.1.2. Farklı doğrultuda düğüm üzerinde karşılıklı çakışma

Bölüm 3.1.2'de ayrıntılı olarak açıklanan farklı doğrultuda düğüm üzerinde karşılıklı çakışma tipi Şekil 3.10'daki sözde kodun 29-34 satırları algoritmik olarak ifade edilmiştir. Bu çakışma tipi bekleme ve rota değiştirme olmak üzere iki temel strateji ile çözülebilmektedir. Önerilen algoritma işlem yapılan OTA'yı çakışma meydana gelen düğümdeki diğer OTA düğümü terk edene kadar bekletir. Bu bekleme işlemi gerçek maliyete satır 5'de hesaplanan bekleme zamanı eklenerek gerçekleştirilir. Bu işlem sonucunda işlem yapılan OTA için çakışma meydana gelen düğüme gelme maliyeti artırılmış olur. Önerilen algoritma çakışma kontrollerini yaptıktan sonra *AçıkListe*'yi tekrardan maliyete göre sıraladığından dolayı en düşük maliyetli düğüm değişebilir. Bu durumda maliyeti güncellenen düğüm hala *AçıkListe*'deki en düşük maliyetli düğüm ise işlem yapılan OTA bekletilmiş olur, değilse rota değiştirmiş olur.

3.2.1.3. Hız farkından kaynaklanan arka arkaya düğüm çakışması

Bölüm 3.1.3'de ayrıntılı olarak açıklanan hız farkından kaynaklanan arka arkaya düğüm çakışması Şekil 3.10'da verilen sözde kodun 36. satırından itibaren algoritmik olarak ifade edilmiştir. Bu çakışma tipinde iki durum meydana gelmektedir;

1. Şekil 3.10'daki sözde kodda 38.satırda ifade edilen ilk durum yavaş olan OTA'nın *KapalıListe*'ye daha önce girmesi durumudur. Bu durumda önerilen algoritma arkadan gelen hızlı OTA'yı çakışma meydana gelen düğümdeki yavaş OTA düğümü terk edene kadar bekletir. Bu bekleme işlemi gerçek maliyete satır 5'te hesaplanan bekleme zamanı eklenerek gerçekleştirilir. Bu işlem sonucunda işlem yapılan OTA için çakışma meydana gelen düğüme gelme maliyeti artırılmış olur. Önerilen algoritma çakışma kontrollerini yaptıktan sonra *AçıkListe*'yi tekrardan maliyete göre sıraladığından dolayı en düşük maliyetli düğüm değişebilir. Bu durumda maliyeti güncellenen düğüm hala *AçıkListe*'deki en düşük maliyetli düğüm ise işlem yapılan OTA bekletilmiş olur, değilse rota değiştirmiş olur.

2. Şekil 3.10'daki sözde kodda 44.satırda ifade edilen ikinci durum hızlı olan OTA'nın *KapalıListe*'ye daha önce girmesi durumudur. Böyle bir durumda yavaş olan OTA'nın rotası zorunlu olarak değiştirilmektedir.

3.2.2. Kenar çakışma kontrolü

İki tip çakışma türünden ikincisi olan kenar çakışması için Şekil 11'de sözde kodu verilen fonksiyon çözüm üretebilmektedir. *ÇakışmaKontrolKenarListesi* gidilen ve gidilmesi muhtemel kenarları tuttuğundan dolayı kenar çakışma kontrolleri *ÇakışmaKontrolKenarListesi* üzerinden yapılmaktadır. *AçıkListe*'den elde edilen aday düğüm ve ebeveyn düğümü arasındaki kenar, *ÇakışmaKontrolKenarListesi*'ndeki *kenarın_gidilme_durumu* özelliği *true* olan kenarlar ile kenar çakışması kontrolü için karşılaştırılır. Kenar üzerinde karşılıklı çakışma ve hız farkından kaynaklanan arka arkaya kenar çakışması olmak üzere iki tip kenar çakışması durumu vardır.

3.2.2.1. Hız farkından kaynaklanan arka arkaya kenar çakışması

Bölüm 3.1.4'de ayrıntılı olarak açıklanan hız farkından kaynaklanan arka arkaya kenar çakışması Şekil 3.11'de verilen sözde kodun 5. satırında algoritmik olarak ifade edilmiştir. Bu çakışma tipinde iki durum meydana gelmektedir.

1. Şekil 3.11'deki sözde kodda 7.satırda ifade edilen ilk durum yavaş olan OTA'nın *ÇakışmaKontrolKenarListesi*'ne daha önce girmesi durumudur. Bu durumda önerilen algoritma arkadan gelen hızlı OTA'yı çakışma meydana gelen kenara giriş düğümünde yavaş OTA kenarın bitiş düğümünü terk edene kadar bekletir. Bu bekleme işlemi gerçek maliyete bekleme zamanı eklenerek gerçekleştirilir (11. satır). Bu işlem sonucunda işlem yapılan OTA için çakışma meydana gelen kenara giriş düğümüne gelme maliyeti artırılmış olur. Önerilen algoritma çakışma kontrollerini yaptıktan sonra *AçıkListe*'yi tekrardan maliyete göre sıraladığından dolayı en düşük maliyetli düğüm değişebilir. Bu durumda maliyeti güncellenen düğüm hala *AçıkListe*'deki en düşük maliyetli düğüm ise işlem yapılan OTA bekletilmiş olur, değilse rota değiştirmiş olur.

```
1.  KenarÇakışmaKontrol(ÇakışmaKontrolKenarListesi, AçıkListe,v)
2.  {
3.      foreach e_j in ÇakışmaKontrolKenarListesi
4.      {
5.          if((v.ebeveyn == e_j.bağlantı_düğümü1 && v== e_j. bağlantı_düğümü2)&&(( v. T_g>= e_j T_g && v. T_g <= e_j T_c)|| (v. T_c >= e_j T_g &&
            v. T_c <= e_j T_c) ||( e_j. T_g>= v. T_g && v. T_c >= e_j. T_g ) || (v. T_g <= e_j .T_c && v. T_c >= e_j .T_c)))
6.          {
7.              if(v. T_g > e_j T_g && v. T_c < e_j T_c && v.hız> e_j.hız)
8.              {
9.                  if(n_j.gidilen_kenar == true)
10.                 {
11.                     v.gercek_maliyet = v.gercek_maliyet +( e_j T_c - v. ebeveyn. T_c );
12.                     v. T_g = v. T_g + ( e_j T_c - v. ebeveyn. T_c );
13.                     v. T_c = v. T_c + ( e_j T_c - v. ebeveyn. T_c );
14.                 }
15.             }
16.             if(v. T_g  <  e_j T_g &&  v. T_c  >  e_j T_c &&  v.hız < e_j.hız)
17.             {
18.                 if(e_j. gidilen_kenar  == true)
19.                 {
20.                     AçıkListe.Kaldır(v);
21.                 }
22.             }
23.         }
24.         if((v.ebeveyn ==  e_j.bağlantı_düğümü1 && v == e_j.bağlantı_düğümü2) && (( v. T_g >=  e_j T_g && v. T_g  <= e_j T_c)|| (v. T_c >= e_j T_g
            && v. T_c <= e_j T_c) ||( e_j T_g>= v. T_g && v. T_c >= e_j T_g ) || (v. T_g <= e_j T_c && v. T_c >= e_j T_c)) )
25.         {
26.             if(n_j.gidilen_kenar == true)
27.             {
28.                 kontrol=0;
29.                 foreach d_k in ÇakışmaKontrolKenarListesi
30.                     if (v.ebeveyn== d_k.bağlantı_düğümü1 && v.ebeveyn.ebeveyn== d_k.bağlantı_düğümü2 && n_j== d_k)
31.                     {
32.                         AçıkListe.Kaldır(v);
33.                         kontrol=1;
34.                         break;
35.                     }
36.                 if  (k<= ÇakışmaKontrolKenarListesi.uzunluk)
                        AçıkListe.Kaldır(v);
37.                 if  (kontrol==0)
38.                 {
39.                     v.gercek_maliyet = v.gercek_maliyet +( e_j T_c - v. ebeveyn. T_c );
40.                     v. T_g = v. T_g + ( e_j T_c - v. ebeveyn. T_c );
41.                     v. T_c = v. T_c + ( e_j T_c - v. ebeveyn. T_c );
42.                 }
43.             }
44.         }
45.     }
46. }
```

Şekil 3.11. Kenar Çakışma Kontrolü Fonksiyonu Sözde (Pseudo) Kodu

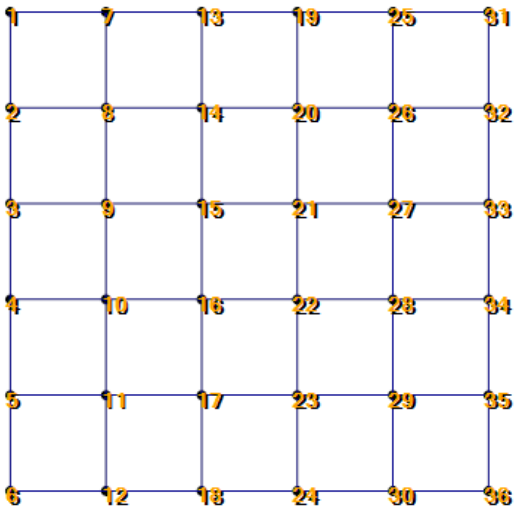
2. Şekil 3.11'deki sözde kodda 16.satırda ifade edilen ikinci durumda *AçıkListe*'deki en düşük maliyetli düğümün sürekli olarak hızlı OTA'ya ait olması durumunda hızlı OTA işlem önceliği elde etmiş olmaktadır. Dolayısıyla hızlı olan OTA'nın gittiği kenarlar sürekli olarak *ÇakışmaKontrolKenarListesi*'ne eklenir. Böyle bir durumda yavaş olan OTA'nın rotası zorunlu olarak değiştirilmektedir.

3.2.2.2. Kenar üzerinde karşılıklı çakışma

Bölüm 3.1.5.'de ayrıntılı olarak açıklanan kenar üzerinde karşılıklı çakışma tipi Şekil 3.11'de verilen sözde kodun 24. satırında algoritmik olarak ifade edilmiştir. Sözde kodun 29. ve 36. satırları arasında *AçıkListe* içerisinde çakışma kontrolü yapılan düğüm (v) ve bu düğümün ebeveyn düğümü arasındaki kenar, *ÇakışmaKontrolKenarListesi* içerisinde çakışma meydana gelen kenar (d_k) karşılaştırılır. Burada d_k kenarı, v düğümü ile ebeveyn düğümü arasındaki kenara eşitse v düğümü *AçıkListe*'den kaldırılır. Bu durumda işlem yapılan OTA'nın çakışmanın olduğu kenara gitme ihtimali ortadan kaldırılmış olur. Bunun dışında kalan durumlarda v düğümünün maliyeti artırılır (39-41. satır arası). Bu durumda maliyeti güncellenen düğüm hala *AçıkListe*'deki en düşük maliyetli düğüm ise işlem yapılan OTA bekletilmiş olur, değilse rota değiştirmiş olur.

3.3. Test Ortamı ve Örnek Çözümler

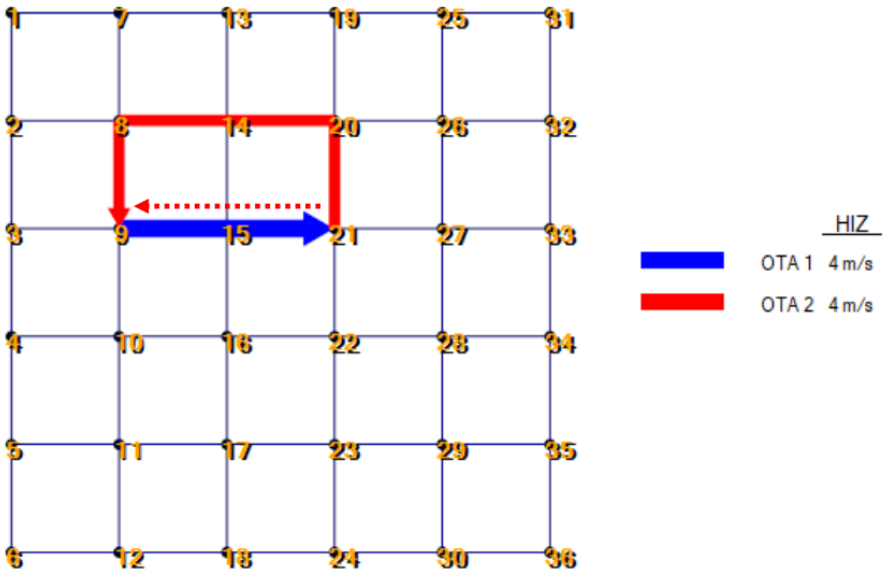
Önerilen algoritmanın testi için olası çakışmaların test edilebileceği ve izlenebileceği Şekil 3.12'de verilen karesel (grid) test ortamı oluşturulmuştur. Komşuluk matrisleri kullanılarak oluşturulan ortamdaki yollar tek şeritli olarak (Aynı anda yolun aynı noktasında yalnızca bir OTA bulunabilir) oluşturulmuştur. Test ortamındaki düğümlerde aynı anda yalnızca bir OTA bulunabilir. OTA'ların düğümlerdeki malzeme yükleme ve boşaltma süreleri sabit bir süre olarak kabul edilmektedir. Farklı hızlara sahip olabilen OTA'ların çalışma süreleri boyunca hızları değişmemektedir.



Şekil 3.12. Test Ortamı

Şekil 3.12'de gösterilen çalışma ortamındaki her iki düğüm arası mesafe 4m olarak belirlenmiştir. Şekilde numaralar düğümleri (yük alma-boşaltma alanlarını), çizgiler ise yolları ifade etmektedir.

Örnek-1 aynı doğrultuda düğüm üzerinde karşılıklı çakışma durumunu ele almaktadır. Bu örnek, hızları aynı ve rotası 9-21 olan OTA_1 ile rotası 21-9 olan OTA_2 için oluşturulmuştur. Örnek-1'in ortam üzerinde gösterimi Şekil 3.13'te verilmiştir. Buna göre 15 numaralı düğümde çakışma meydana gelmektedir. Bu çakışma tipi Bölüm 3.2.1.1'de açıklanan duruma karşılık gelmektedir. Önerilen algoritma çakışmayı önlemek için OTA_2 'nin rotasını değiştirmiştir.



Şekil 3.13. İki OTA İçin Aynı Doğrultuda Düğüm Üzerinde Karşılıklı Çakışma

Şekil 3.13'te OTA_2 'nin çakışma olmaması durumunda gideceği rota kesik çizgili ok ile gösterilirken önerilen algoritmanın oluşturduğu rota düz okla gösterilmiştir. Çizelge 3.2 tüm OTA'lar için önerilen algoritmanın bulduğu rotaları ve bu rotalar üzerindeki her düğüme giriş-çıkış zamanlarını göstermektedir. Çizelge 3.3 her OTA için çakışmaların dikkate alındığı ve alınmadığı rota uzunlukları ile rota sürelerini göstermektedir. Buna göre çakışmalar dikkate alınmadan hesaplanan toplam rota uzunluğu 16m, süre 4.04s iken, önerilen algoritmanın çakışmaları çözerek bulduğu rota uzunluğu 24m, süre 6.06s'dir. Burada çakışmadan kaynaklı olarak rota uzunluğu ve rota süresi artmıştır. Fakat önerilen

algoritmanın çakışma durumunu hesaba katmadan rota oluşturduğu varsayılsaydı, OTA'ların hareketi sırasında herhangi bir çakışma meydana geldiğinde yani OTA'lardan ikisi de 15 numaralı düğüme gelmek istediğinde oluşacak çakışmadan dolayı OTA'lardan biri diğerinin hareketinin bitmesini beklemek zorunda kalacaktı. Örnek-1'de her iki OTA'da karşılıklı olarak birbirinin yolundan gitmek isteyeceği için burada kilitlenme durumu meydana gelecekti. Özellikle çoklu OTA'ların hareket ettiği ortamda OTA'lar anlık çakışma durumuna göre birbirini beklemek zorunda kaldıklarında düğüm üzerinde tıkanıklıklar meydana gelebilir. Ayrıca her iki OTA'da karşılıklı birbirini beklerse bu kilitlenme durumunu ortaya çıkabilir. Başlangıçta çakışmasız rotalar hesaplandığında algoritma her bir OTA'yı mümkün olan uygun rotalardan götürerek tıkanıklıkları ve kilitlenmeleri önlemektedir.

Çizelge 3.2. Örnek-1 için OTA_1 ve OTA_2'nin Düğümlere Giriş Çıkış Zamanları

OTA	$DÜGÜM^{Dügüme\_gelis\_zamani(t_g)}_{Dügümden\_cikis\_zamani(t_c)}$
OTA_1	$9^{(0.00)}_{(0.01)} \rightarrow 15^{(1.01)}_{(1.02)} \rightarrow 21^{(2.02)}_{(-)}$
OTA_2	$21^{(0.00)}_{(0.01)} \rightarrow 20^{(1.01)}_{(1.02)} \rightarrow 14^{(2.02)}_{(2.03)} \rightarrow 8^{(3.03)}_{(3.04)} \rightarrow 9^{(4.04)}_{(-)}$

Çizelge 3.3. Örnek-1 için OTA_1 ve OTA_2'nin Çakışmalı/Çakışmasız Rota Uzunluğu ve Rota Tamamlanma Süreleri

OTA	**Çakışmanın Dikkate Alınmadığı Rota Uzunluğu**	**Çakışmanın Dikkate Alındığı Rota Uzunluğu**	**Çakışmanın Dikkate Alınmadığı Rota Süresi**	**Çakışmanın Dikkate Alındığı Rota süresi**
OTA_1	8m	8m	2.02s	2.02s
OTA_2	8m	16m	2.02s	4.04s
Maksimum	*8m*	*16m*	*2.02s*	*4.04s*

Çizelge 3.4'te Örnek-1'in önerilen algoritmayla gerçeklenmesi gösterilmektedir. OTA_1 R_1 şeklinde ifade edilip, hızı 4m/sn'dir. OTA_1'in rotasının başlangıç düğümü 9 numaralı düğüm, bitiş düğümü ise 21 numaralı düğümdür. OTA_2 R_2 şeklinde ifade edilip, hızı 4m/sn'dir. OTA_2'nin rotasının başlangıç düğümü 21 numaralı düğüm, bitiş düğümü ise 9 numaralı düğümdür. Her bir düğüm için OTA'ların bekleme süresi 0.01sn olarak belirlenmiştir.

Çizelge 3.4. Örnek-1 için Önerilen Algoritmanın Çalışma Şeklinin Adım Adım Gösterimi

#	KapalıListe	Açıkliste	ÇakışmaKontrol KenarListesi	**Açıklama:** Düğüm çakışmaları *KapalıListe* üzerinden kontrol ediliyor. Kenar çakışmaları ise *ÇakışmaKontrolKenarListesi* üzerinden kontrol ediliyor.
1		$9^{\text{R1-root}}$(0+2=2), $21^{\text{R2-root}}$(0+2=2)		
2	$9^{(R1-root)}_{(0.00-0.01)}$	$21^{\text{R2-root}}$(0+2=2), $10^{\text{R1-9}}$(1.01+3=4.01), $8^{\text{R1-9}}$(1.01+3=4.01), $15^{\text{R1-9}}$(1.01+1=2.01), $3^{\text{R1-9}}$(1.01+3=4.01),	$9-10^{R1}_{(0.01-1.01)}$, $9-15^{R1}_{(0.01-1.01)}$, $9-3^{R1}_{(0.01-1.01)}$, $9-8^{R1}_{(0.01-1.01)}$,	
3	$9^{(R1-root)}_{(0.00-0.01)}$, $21^{(R2-root)}_{(0.00-0.01)}$	$10^{\text{R1-9}}$(1.01+3=4.01), $8^{\text{R1-9}}$(1.01+3=4.01), $15^{\text{R1-9}}$(1.01+1=2.01), $3^{\text{R1-9}}$(1.01+3=4.01), $22^{\text{R2-21}}$(1.01+3=4.01), $20^{\text{R2-21}}$(1.01+3=4.01), $27^{\text{R2-21}}$(1.01+3=4.01), $15^{\text{R2-21}}$(1.01+1=2.01)	$9-10^{R1}_{(0.01-1.01)}$, $9-15^{R1}_{(0.01-1.01)}$, $9-3^{R1}_{(0.01-1.01)}$, $9-8^{R1}_{(0.01-1.01)}$, $21-22^{R2}_{(0.01-1.01)}$, $21-20^{R2}_{(0.01-1.01)}$, $21-27^{R2}_{(0.01-1.01)}$, $21-15^{R2}_{(0.01-1.01)}$,	
4	$9^{(R1-root)}_{(0.00-0.01)}$, $21^{(R2-root)}_{(0.00-0.01)}$, $15^{(R1-9)}_{(1.01-1.02)}$	$10^{\text{R1-9}}$(1.01+3=4.01), $8^{\text{R1-9}}$(1.01+3=4.01), $3^{\text{R1-9}}$(1.01+3=4.01), $22^{\text{R2-21}}$(1.01+3=4.01), $20^{\text{R2-21}}$(1.01+3=4.01), $27^{\text{R2-21}}$(1.01+3=4.01), $15^{\text{R2-21}}$(1.01+1=2.01), $16^{\text{R1-15}}$(2.02+2=4.02), $14^{\text{R1-15}}$(2.02+2=4.02), $21^{\text{R1-15}}$(2.02+0=2.02)	$9-10^{R1}_{(0.01-1.01)}$, $9-15^{R1}_{(0.01-1.01)}$, $9-3^{R1}_{(0.01-1.01)}$, $9-8^{R1}_{(0.01-1.01)}$, $21-22^{R2}_{(0.01-1.01)}$, $21-20^{R2}_{(0.01-1.01)}$, $21-27^{R2}_{(0.01-1.01)}$, $21-15^{R2}_{(0.01-1.01)}$,	R1 ile R2 OTA'ları arasında 15 numaralı düğümde aynı doğrultuda düğüm üzerinde karşılıklı çakışma durumu meydana gelmektedir. R2 OTA'sının 15 numaralı

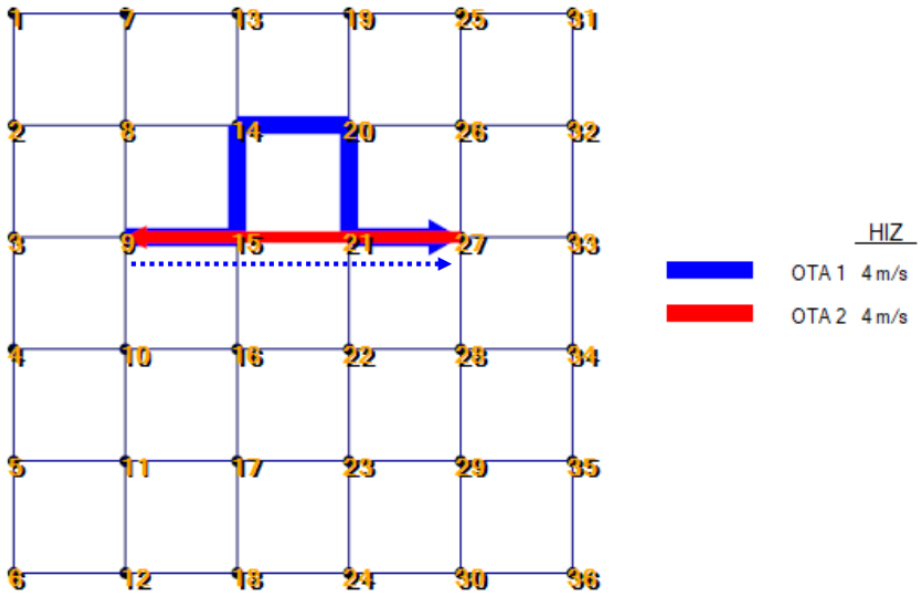
			$15-16^{R1}_{(1.02-2.02)}$, $15-14^{R1}_{(1.02-2.02)}$, $15-21^{R1}_{(1.02-2.02)}$,	düğümü işgal etme saniyesi (1.01-1.02) ile R1 OTA'sının 15 numaralı düğümü işgal etme saniyesi (1.01-1.02) çakışmaktadır. Bu çakışma tipinde R2'ye ait 15 numaralı düğüm (kırmızı dikdörtgen ile gösterilmiştir)AçıkListe'den kaldırımıştır. Böylece R2'nin rotası değiştirilerek çakışma önlenmiştir.
5	$9^{(R1-root)}_{(0.00-0.01)}$, $21^{(R2-root)}_{(0.00-0.01)}$, $15^{(R1-9)}_{(1.01-1.02)}$, $21^{(R1-15)}_{(2.02-2.03)}(biti\text{Ş})$	~~10^{R1-9}(1.01+3=4.01),~~ ~~8^{R1-9}(1.01+3=4.01),~~ ~~3^{R1-9}(1.01+3=4.01),~~ 22^{R2-21}(1.01+3=4.01), 20^{R2-21}(1.01+3=4.01), 27^{R2-21}(1.01+3=4.01), ~~16^{R1-15}(2.02+2=4.02),~~ ~~14^{R1-15}(2.02+2=4.02),~~ ~~22^{R1-21}(3.03+1=4.03),~~ ~~20^{R1-21}(3.03+1=4.03),~~ ~~27^{R1-21}(3.03+1=4.03)~~	$9-10^{R1}_{(0.01-1.01)}$, $9-15^{R1}_{(0.01-1.01)}$, $9-3^{R1}_{(0.01-1.01)}$, $9-8^{R1}_{(0.01-1.01)}$, $21-22^{R2}_{(0.01-1.01)}$, $21-20^{R2}_{(0.01-1.01)}$, $21-27^{R2}_{(0.01-1.01)}$, $21-15^{R2}_{(0.01-1.01)}$, $15-16^{R1}_{(1.02-2.02)}$, $15-14^{R1}_{(1.02-2.02)}$, $15-21^{R1}_{(1.02-2.02)}$, $21-22^{R1}_{(2.03-3.03)}$, $21-20^{R1}_{(2.03-3.03)}$, $21-27^{R1}_{(2.03-3.03)}$,	R1 OTA'sı bitiş düğümüne geldiği zaman R1 OTA'na ait tüm düğümler AçıkListeden kaldırılır.
6	$9^{(R1-root)}_{(0.00-0.01)}$, $21^{(R2-root)}_{(0.00-0.01)}$, $15^{(R1-9)}_{(1.01-1.02)}$, $21^{(R1-15)}_{(2.02-2.03)}$(bitis)	22^{R2-21}(1.01+3=4.01), 27^{R2-21}(1.01+3=4.01), 19^{R2-20}(2.02+4=6.02),	$9-10^{R1}_{(0.01-1.01)}$, $9-15^{R1}_{(0.01-1.01)}$, $9-3^{R1}_{(0.01-1.01)}$,	

	$\mathbf{20}^{(\boldsymbol{R2-21})}_{\mathbf{(1.01-1.02)}}$,	$26^{\text{R2-20}}$(2.02+4=6.02), $14^{\text{R2-20}}$(2.02+2=4.02)	$9-8^{R1}_{(0.01-1.01)}$, $21-22^{R2}_{(0.01-1.01)}$, $21-20^{R2}_{(0.01-1.01)}$, $21-27^{R2}_{(0.01-1.01)}$, $21-15^{R2}_{(0.01-1.01)}$, $15-16^{R1}_{(1.02-2.02)}$, $15-14^{R1}_{(1.02-2.02)}$, $15-21^{R1}_{(1.02-2.02)}$, $21-22^{R1}_{(2.03-3.03)}$, $21-20^{R1}_{(2.03-3.03)}$, $21-27^{R1}_{(2.03-3.03)}$, $20-19^{R2}_{(1.02-2.02)}$, $20-26^{R2}_{(1.02-2.02)}$, $20-14^{R2}_{(1.02-2.02)}$,	
	⋮	⋮	⋮	⋮
Algoritma bu şekilde devam etmektedir.				

Çizelge 3.4'te Örnek-1'in önerilen algoritma üzerinde adım adım gerçeklenmesi esnasında *KapalıListe*, *AçıkListe* ve *ÇakışmaKontrolKenarListesi*'nin her adımdaki durumu yer almaktadır. 4. adımda çakışma durumu meydana gelmektedir ve meydana gelen çakışmanın nasıl çözüldüğü Çizelge 3.4'te gösterilmektedir. İlk altı adımdan sonraki adımlar benzer şekilde devam edeceğinden çizelgede sadece ilk altı adım verilmiştir. Algoritmanın ilk adımında tüm OTA'ların başlangıç düğümleri *AçıkListe*'ye eklenir.

Örnek-2 kenar üzerinde karşılıklı çakışma durumunu ele almaktadır. Bu örnek, rotası 9-27 olan OTA_1 ile rotası 27-9 olan OTA_2 için oluşturulmuştur. Örnek-2'nin ortam üzerinde gösterimi Şekil 3.14'te verilmiştir. Buna göre 15 ile 21 numaralı düğümler

arasındaki kenarda çakışma meydana gelmektedir. Bu çakışma tipi Bölüm 3.2.2.2'de açıklanan duruma karşılık gelmektedir. Önerilen algoritma çakışmayı önlemek için OTA_1'in rotasını değiştirmiştir. Şekil 3.14'te OTA_1'in çakışma olmaması durumunda gideceği rota kesik çizgili ok ile gösterilirken önerilen algoritmanın oluşturduğu rota düz okla gösterilmiştir.



Şekil 3.14. İki OTA İçin Aynı Doğrultuda Düğüm Üzerinde Karşılıklı Çakışma

Çizelge 3.5. Örnek-2 için OTA_1 ve OTA_2'nin Düğümlere Giriş Çıkış Zamanları

OTA	$DÜGÜM^{Dügüme\_gelis\_zamani(t_g)}_{Dügümden\_cikis\_zamani(t_c)}$
OTA_1	$9^{(0.00)}_{(0.01)} \rightarrow 15^{(1.01)}_{(1.02)} \rightarrow 14^{(2.02)}_{(2.03)} \rightarrow 20^{(3.03)}_{(3.04)} \rightarrow 21^{(4.04)}_{(4.05)} \rightarrow 27^{(5.05)}_{(-)}$
OTA_2	$27^{(0.00)}_{(0.01)} \rightarrow 21^{(1.01)}_{(1.02)} \rightarrow 15^{(2.02)}_{(2.03)} \rightarrow 9^{(3.03)}_{(-)}$

Çizelge 3.6. Örnek-2 için OTA_1 ve OTA_2'nin Çakışmalı/Çakışmasız Rota Uzunluğu ve Rota Tamamlanma Süreleri

OTA	Çakışmanın Dikkate Alınmadığı Rota Uzunluğu	Çakışmanın Dikkate Alındığı Rota Uzunluğu	Çakışmanın Dikkate Alınmadığı Rota Süresi	Çakışmanın Dikkate Alındığı Rota süresi
OTA_1	12m	20m	3.03s	5.05s
OTA_2	12m	12m	3.03s	3.03s
Maksimum	*12m*	*12m*	*3.03s*	*5.05s*

Çizelge 3.7. Örnek-2 için Önerilen Algoritmanın Çalışma Şeklinin Adım Adım Gösterimi

	KapalıListe	**Açıkliste**	**ÇakışmaKontrolKenarListesi**	**Açıklama:** Düğüm çakışmaları *KapalıListe* üzerinden kontrol ediliyor. Kenar çakışmaları ise *ÇakışmaKontrolKenarListesi* üzerinden kontrol ediliyor.
1		$9^{\text{R1-root}}(0+3=3)$, $27^{\text{R2-root}}(0+3=3)$		
2	$9^{(R1-root)}_{(0.00-0.01)}$	$27^{\text{R2-root}}(0+3=3)$, $10^{\text{R1-9}}(1.01+4=5.01)$, $8^{\text{R1-9}}(1.01+4=5.01)$, $15^{\text{R1-9}}(1.01+2=3.01)$, $3^{\text{R1-9}}(1.01+4=5.01)$,	$9-10^{R1}_{(0.01-1.01)}$, $9-8^{R1}_{(0.01-1.01)}$, $9-15^{R1}_{(0.01-1.01)}$, $9-3^{R1}_{(0.01-1.01)}$,	
3	$9^{(R1-root)}_{(0.00-0.01)}$, $27^{(R2-root)}_{(0.00-0.01)}$	$10^{\text{R1-9}}(1.01+3=4.01)$, $8^{\text{R1-9}}(1.01+3=4.01)$, $15^{\text{R1-9}}(1.01+1=2.01)$, $3^{\text{R1-9}}(1.01+3=4.01)$, $28^{\text{R2-21}}(1.01+4=5.01)$, $26^{\text{R2-21}}(1.01+4=5.01)$, $33^{\text{R2-21}}(1.01+4=5.01)$, $21^{\text{R2-21}}(1.01+2=3.01)$	$9-10^{R1}_{(0.01-1.01)}$, $9-8^{R1}_{(0.01-1.01)}$, $9-15^{R1}_{(0.01-1.01)}$, $9-3^{R1}_{(0.01-1.01)}$, $27-28^{R2}_{(0.01-1.01)}$, $27-26^{R2}_{(0.01-1.01)}$, $27-33^{R2}_{(0.01-1.01)}$, $27-21^{R2}_{(0.01-1.01)}$,	
4	$9^{(R1-root)}_{(0.00-0.01)}$, $27^{(R2-root)}_{(0.00-0.01)}$, $15^{(R1-9)}_{(1.01-1.02)}$	$10^{\text{R1-9}}(1.01+3=4.01)$, $8^{\text{R1-9}}(1.01+3=4.01)$, $3^{\text{R1-9}}(1.01+3=4.01)$, $28^{\text{R2-21}}(1.01+4=5.01)$, $26^{\text{R2-21}}(1.01+4=5.01)$, $33^{\text{R2-21}}(1.01+4=5.01)$, $21^{\text{R2-21}}(1.01+2=3.01)$, $16^{\text{R1-15}}(2.02+3=5.02)$, $14^{\text{R1-15}}(2.02+3=5.02)$, $21^{\text{R1-15}}(2.02+1=3.02)$,	$9-10^{R1}_{(0.01-1.01)}$, $9-8^{R1}_{(0.01-1.01)}$, $9-15^{R1}_{(0.01-1.01)}$, $9-3^{R1}_{(0.01-1.01)}$, $27-28^{R2}_{(0.01-1.01)}$, $27-26^{R2}_{(0.01-1.01)}$, $27-33^{R2}_{(0.01-1.01)}$, $27-21^{R2}_{(0.01-1.01)}$, $15-16^{R1}_{(1.02-2.02)}$, $15-14^{R1}_{(1.02-2.02)}$, $15-21^{R1}_{(1.02-2.02)}$,	
5	$9^{(R1-root)}_{(0.00-0.01)}$, $27^{(R2-root)}_{(0.00-0.01)}$, $15^{(R1-9)}_{(1.01-1.02)}$,	$10^{\text{R1-9}}(1.01+3=4.01)$, $8^{\text{R1-9}}(1.01+3=4.01)$, $3^{\text{R1-9}}(1.01+3=4.01)$, $28^{\text{R2-21}}(1.01+4=5.01)$,	$9-10^{R1}_{(0.01-1.01)}$, $9-8^{R1}_{(0.01-1.01)}$, $9-15^{R1}_{(0.01-1.01)}$,	

	$21^{(R2-27)}_{(1.01-1.02)}$,	$26^{R2\text{-}21}$(1.01+4=5.01), $33^{R2\text{-}21}$(1.01+4=5.01), $16^{R1\text{-}15}$(2.02+3=5.02), $14^{R1\text{-}15}$(2.02+3=5.02), $21^{R1\text{-}15}$(2.02+1=3.02), $22^{R2\text{-}21}$(2.02+3=5.02), $20^{R2\text{-}21}$(2.02+3=5.02), $15^{R2\text{-}21}$(2.02+1=3.02)	$9-3^{R1}_{(0.01-1.01)}$, $27-28^{R2}_{(0.01-1.01)}$, $27-26^{R2}_{(0.01-1.01)}$, $27-33^{R2}_{(0.01-1.01)}$, $27-21^{R2}_{(0.01-1.01)}$, $15-16^{R1}_{(1.02-2.02)}$, $15-14^{R1}_{(1.02-2.02)}$, $15-21^{R1}_{(1.02-2.02)}$, $21-22^{R2}_{(0.01-1.01)}$, $21-20^{R2}_{(0.01-1.01)}$, $21-15^{R2}_{(0.01-1.01)}$,	
6	$9^{(R1-root)}_{(0.00-0.01)}$, $27^{(R2-root)}_{(0.00-0.01)}$, $15^{(R1-9)}_{(1.01-1.02)}$, $21^{(R2-27)}_{(1.01-1.02)}$, $15^{(R2-21)}_{(2.02-2.03)}$,	$10^{R1\text{-}9}$(1.01+3=4.01), $8^{R1\text{-}9}$(1.01+3=4.01), $3^{R1\text{-}9}$(1.01+3=4.01), $28^{R2\text{-}21}$(1.01+4=5.01), $26^{R2\text{-}21}$(1.01+4=5.01), $33^{R2\text{-}21}$(1.01+4=5.01), $16^{R1\text{-}15}$(2.02+3=5.02), $14^{R1\text{-}15}$(2.02+3=5.02), $21^{R1\text{-}15}$(2.02+1=3.02), $22^{R2\text{-}21}$(2.02+3=5.02), $20^{R2\text{-}21}$(2.02+3=5.02), $16^{R2\text{-}15}$(3.03+2=5.03), $14^{R2\text{-}15}$(3.03+2=5.03), $9^{R2\text{-}15}$(3.03+0=3.03),	$9-10^{R1}_{(0.01-1.01)}$, $9-8^{R1}_{(0.01-1.01)}$, $9-15^{R1}_{(0.01-1.01)}$, $9-3^{R1}_{(0.01-1.01)}$, $27-28^{R2}_{(0.01-1.01)}$, $27-26^{R2}_{(0.01-1.01)}$, $27-33^{R2}_{(0.01-1.01)}$, $27-21^{R2}_{(0.01-1.01)}$, $15-16^{R1}_{(1.02-2.02)}$, $15-14^{R1}_{(1.02-2.02)}$, $15-21^{R1}_{(1.02-2.02)}$, $21-22^{R2}_{(1.02-2.02)}$, $21-20^{R2}_{(1.02-2.02)}$, $21-15^{R2}_{(1.02-2.02)}$, $15-16^{R2}_{(2.03-3.03)}$, $15-14^{R2}_{(2.03-3.03)}$, $15-9^{R2}_{(2.03-3.03)}$,	R1 ile R2 OTA'ları arasında 15-21 kenarında kenar üzerinde karşılıklı çakışma durumu meydana gelmektedir. R2 OTA'sının 21-15 kenarını işgal etme saniyesi (1.02-2.02) ile R1 OTA'sının 15-21 kenarını işgal etme saniyesi (1.02-2.02) çakışmaktadır. Bu çakışma tipinde R1'e ait 21 numaralı düğüm (mavi dikdörtgen ile gösterilmiştir)AçıkListe'den kaldırımıştır. Böylece R1'in rotası değiştirilerek çakışma önlenmiştir.
7	$9^{(R1-root)}_{(0.00-0.01)}$, $27^{(R2-root)}_{(0.00-0.01)}$, $15^{(R1-9)}_{(1.01-1.02)}$, $21^{(R2-27)}_{(1.01-1.02)}$, $15^{(R2-21)}_{(2.02-2.03)}$, $9^{(R2-15)}_{(3.03-3.04)}(bitiş)$	$10^{R1\text{-}9}$(1.01+3=4.01), $8^{R1\text{-}9}$(1.01+3=4.01), $3^{R1\text{-}9}$(1.01+3=4.01), ~~$28^{R2\text{-}21}$(1.01+4=5.01),~~ ~~$26^{R2\text{-}21}$(1.01+4=5.01),~~ ~~$33^{R2\text{-}21}$(1.01+4=5.01),~~ $16^{R1\text{-}15}$(2.02+3=5.02), $14^{R1\text{-}15}$(2.02+3=5.02), ~~$22^{R2\text{-}21}$(2.02+3=5.02),~~ ~~$20^{R2\text{-}21}$(2.02+3=5.02),~~ ~~$16^{R2\text{-}15}$(3.03+2=5.03),~~	$9-10^{R1}_{(0.01-1.01)}$, $9-8^{R1}_{(0.01-1.01)}$, $9-15^{R1}_{(0.01-1.01)}$, $9-3^{R1}_{(0.01-1.01)}$, $27-28^{R2}_{(0.01-1.01)}$, $27-26^{R2}_{(0.01-1.01)}$, $27-33^{R2}_{(0.01-1.01)}$, $27-21^{R2}_{(0.01-1.01)}$, $15-16^{R1}_{(1.02-2.02)}$,	R2 OTA'sı bitiş düğümüne geldiği zaman R1 OTA'na ait tüm düğümler AçıkListeden kaldırılır.

		~~$14^{R2\text{-}15}(3.03+2=5.03)$,~~ ~~$10^{R2\text{-}9}(4.04+1=5.04)$,~~ ~~$8^{R2\text{-}9}(4.04+1=5.04)$,~~ ~~$3^{R2\text{-}9}(4.04+1=5.04)$,~~	$15-14^{R1}_{(1.02-2.02)}$, $15-21^{R1}_{(1.02-2.02)}$, $21-22^{R2}_{(1.02-2.02)}$, $21-20^{R2}_{(1.02-2.02)}$, $21-15^{R2}_{(1.02-2.02)}$, $15-16^{R2}_{(2.03-3.03)}$, $15-14^{R2}_{(2.03-3.03)}$, $15-9^{R2}_{(2.03-3.03)}$, $9-10^{R2}_{(3.04-4.04)}$, $9-8^{R2}_{(3.04-4.04)}$, $9-3^{R2}_{(3.04-4.04)}$,	
	⋮	⋮	⋮	⋮
	Algoritma bu şekilde devam etmektedir.			

Çizelge 3.5 tüm OTA'lar için önerilen algoritmanın bulduğu rotaları ve bu rotalar üzerindeki her düğüme giriş-çıkış zamanlarını göstermektedir. Çizelge 3.6 her OTA için çakışmaların dikkate alındığı ve alınmadığı rota uzunlukları ile rota sürelerini göstermektedir. Buna göre çakışmalar dikkate alınmadan hesaplanan toplam rota uzunluğu 24m, süre 6.06s iken, önerilen algoritmanın çakışmaları çözerek bulduğu rota uzunluğu 32m, süre 8.08s'dir. Burada çakışmadan kaynaklı olarak rota uzunluğu ve rota süresi artmıştır. Fakat algoritma başlangıçta çakışmasız rotalar oluşturmasaydı çakışma durumunda iki OTA'da 15-21 kenarında karşı düğüme geçmek için birbirini bekleyecekti ve bu durumda da ortamda kilitlenme meydana gelecekti. Çakışma durumunda OTA'lardan biri diğerinin rotasının tamamlanmasını bekleseydi yani OTA_1 OTA_2 'nin rotasını tamamlamasını bekleseydi, OTA_1 3.04. saniyede harekete başlayacaktı ve 6.06s'de rotasını tamamlayacaktı. Önerilen algoritmanın OTA_1 için çakışmasız olarak hesapladığı süre ise 5.05s dir. Bu durumda önerilen algoritmanın hesapladığı çakışmasız rota daha avantajlıdır.

Çizelge 3.7'de Örnek-2'nin önerilen algoritmayla gerçeklenmesi gösterilmektedir. OTA_1 R_1 şeklinde ifade edilip, hızı 4m/sn'dir. OTA_1 'in rotasının başlangıç düğümü 9 numaralı düğüm, bitiş düğümü ise 27 numaralı düğümdür. OTA_2 R_2 şeklinde ifade edilip, hızı 4m/sn'dir. OTA_2 'nin rotasının başlangıç düğümü 27 numaralı düğüm, bitiş düğümü ise 9 numaralı düğümdür. Her bir düğüm için OTA'ların bekleme süresi 0.01sn olarak

belirlenmiştir. Çizelge 3.7'de Örnek-2'nin önerilen algoritma üzerinde adım adım gerçeklenmesi esnasında *KapalıListe*, *AçıkListe* ve *ÇakışmaKontrolKenarListesi*'nin her adımdaki durumu yer almaktadır. 5. adımda çakışma durumu meydana gelmektedir ve meydana gelen çakışmanın nasıl çözüldüğü Çizelge 3.7'de gösterilmektedir. İlk yedi adımdan sonraki adımlar benzer şekilde devam edeceğinden çizelgede sadece ilk yedi adım verilmiştir.

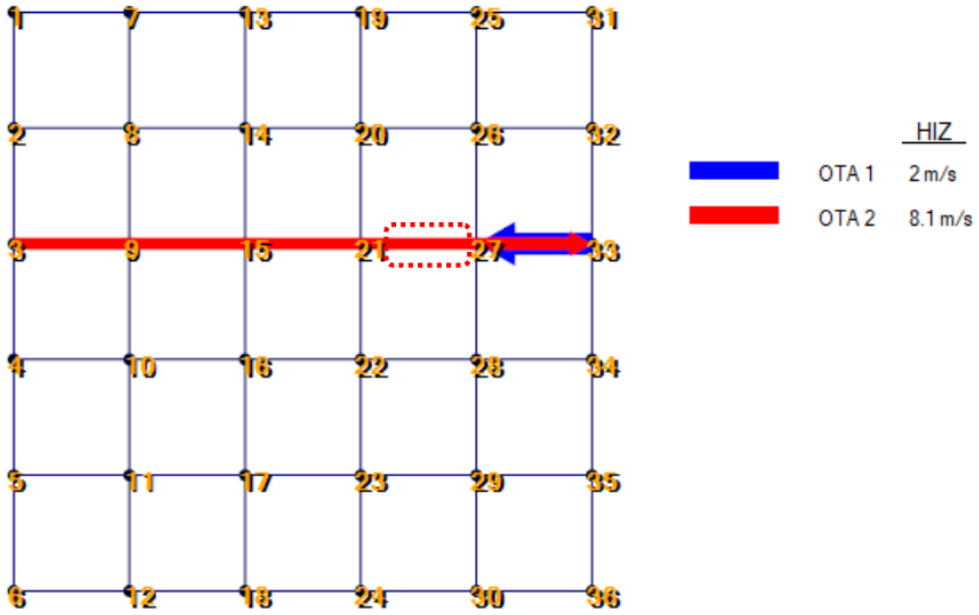
4. BULGULAR VE TARTIŞMA

Önerilen algoritma farklı sayıda heterojen araç için test edilmiştir. Testler bölüm 3.3'te oluşturulan ortam üzerinde yapılmıştır. İlk alt bölümde iki OTA için meydana gelebilecek çakışma tipleri ile ilgili testler yer almaktadır. İkinci alt bölümdeki testler ise on OTA için yapılmıştır ve çok sayıda farklı tipteki çakışma durumlarının meydana geldiği test senaryoları önerilen algoritma üzerinde denenmiştir. Yapılan testlerde OTA'ların başlangıç ve bitiş düğümlerinin önceden bilindiği (çizelgelemenin yapıldığı) varsayılmıştır. Teslerde meydana gelen çakışmalar duruma uygun çakışma çözme stratejisi ile çözülmüştür. Test çalışmaları Windows 10 işletim sistemi ile Visual C# programlama dili kullanılarak gerçekleştirilmiştir.

4.1. İki OTA için Oluşturulan Testler

Test-1 herhangi bir OTA'nın bitiş düğümünde meydana gelen aynı doğrultuda düğüm üzerinde karşılıklı çakışma durumunu ele almaktadır. Bu test, rotası 33-27 olan OTA_1 ile rotası 3-33 olan OTA_2 için oluşturulmuştur. Test-1'nin ortam üzerinde gösterimi Şekil 4.1'de verilmiştir. Buna göre OTA_1'in bitiş düğümü olan 27 numaralı düğümde çakışma meydana gelmektedir. Bu çakışma tipi Bölüm 3.2.1.1'de açıklanan duruma karşılık gelmektedir. Önerilen algoritma çakışmayı önlemek için OTA_2 'yi 21-27 kenarı arasında (Şekil 4.1'de kırmızı kesik çizgili dikdörtgen ile gösterilmiştir) OTA_1 27 numaralı düğümü terk edene kadar bekletmiştir.

Çizelge 4.1 tüm OTA'lar için önerilen algoritmanın bulduğu rotaları ve bu rotalar üzerindeki her düğüme giriş-çıkış zamanlarını göstermektedir. Çizelge 4.2 her OTA için çakışmaların dikkate alındığı ve alınmadığı rota uzunlukları ile rota sürelerini göstermektedir. Buna göre çakışmalar dikkate alınmadan hesaplanan toplam rota uzunluğu 24m, süre 4.527s iken, önerilen algoritmanın çakışmaları çözerek bulduğu rota uzunluğu 24m, süre 4.543s'dir.



Şekil 4.1. İki OTA İçin Aynı Doğrultuda Düğüm üzerinde Karşılıklı Çakışma

Çizelge 4.1. Test-1 için OTA_1 ve OTA_2 'nin Düğümlere Giriş Çıkış Zamanları ve Toplam Rota Uzunlukları

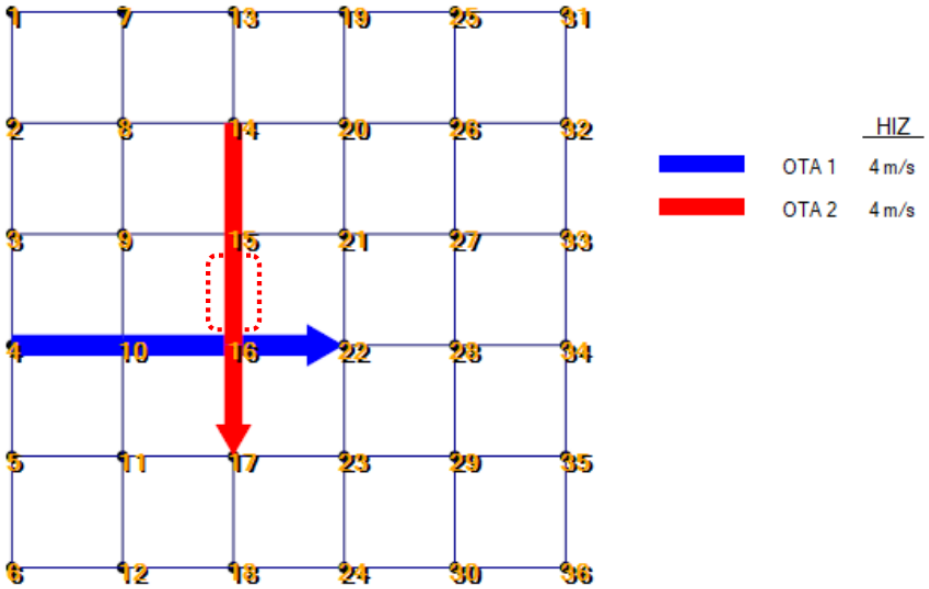
OTA	$DÜGÜM^{Düğüme\_gelis\_zamani(t_g)}_{Düğümden\_cikis\_zamani(t_c)}$
OTA_1	$33^{(0.00)}_{(0.01)} \rightarrow 27^{(2.01)}_{(-)}$
OTA_2	$3^{(0.000)}_{(0.010)} \rightarrow 9^{(0.503)}_{(0.513)} \rightarrow 15^{(1.007)}_{(1.017)} \rightarrow 21^{(1.511)}_{(1.521)} \rightarrow 27^{(2.030)}_{(2.040)} \rightarrow 33^{(2.533)}_{(-)}$

Çizelge 4.2. Test-1 için OTA_1 ve OTA_2 'nin Çakışmalı/Çakışmasız Rota Uzunluğu ve Rota Tamamlanma Süreleri

OTA	**Çakışmanın Dikkate Alınmadığı Rota Uzunluğu**	**Çakışmanın Dikkate Alındığı Rota Uzunluğu**	**Çakışmanın Dikkate Alınmadığı Rota Süresi**	**Çakışmanın Dikkate Alındığı Rota süresi**
OTA_1	4m	4m	2.01s	2.01s
OTA_2	20m	20m	2.517s	2.533s
Maksimum	*20m*	*20m*	*2.517s*	*42.533s*

Test-2 farklı doğrultuda düğüm üzerinde karşılıklı çakışma durumunu ele almaktadır. Bu test, rotası 4-22 olan OTA_1 ile rotası 14-17 olan OTA_2 için oluşturulmuştur. Test-3'ün ortam üzerinde gösterimi Şekil 4.3'te verilmiştir. Buna göre 16 numaralı düğümde çakışma meydana gelmektedir. Bu çakışma tipi Bölüm 3.2.1.2'de açıklanan duruma karşılık

gelmektedir. Önerilen algoritma çakışmayı önlemek için OTA_2'yi 15-16 kenarında (Şekil 4.2'de kırmızı kesik çizgili dikdörtgen ile gösterilmiştir) OTA_1 16 numaralı düğümü terk edene kadar bekletmiştir. Çizelge 4.3 tüm OTA'lar için önerilen algoritmanın bulduğu rotaları ve bu rotalar üzerindeki her düğüme giriş-çıkış zamanlarını göstermektedir. Çizelge 4.4 her OTA için çakışmaların dikkate alındığı ve alınmadığı rota uzunlukları ile rota sürelerini göstermektedir. Buna göre çakışmalar dikkate alınmadan hesaplanan toplam rota uzunluğu 24m, süre 6.06s iken, önerilen algoritmanın çakışmaları çözerek bulduğu rota uzunluğu 24m, süre 6.08s'dir.



Şekil 4.2. İki OTA İçin Farklı Doğrultuda Düğüm üzerinde Karşılıklı Çakışma

Çizelge 4.3. Test-2 için OTA_1 ve OTA_2'nin Düğümlere Giriş Çıkış Zamanları

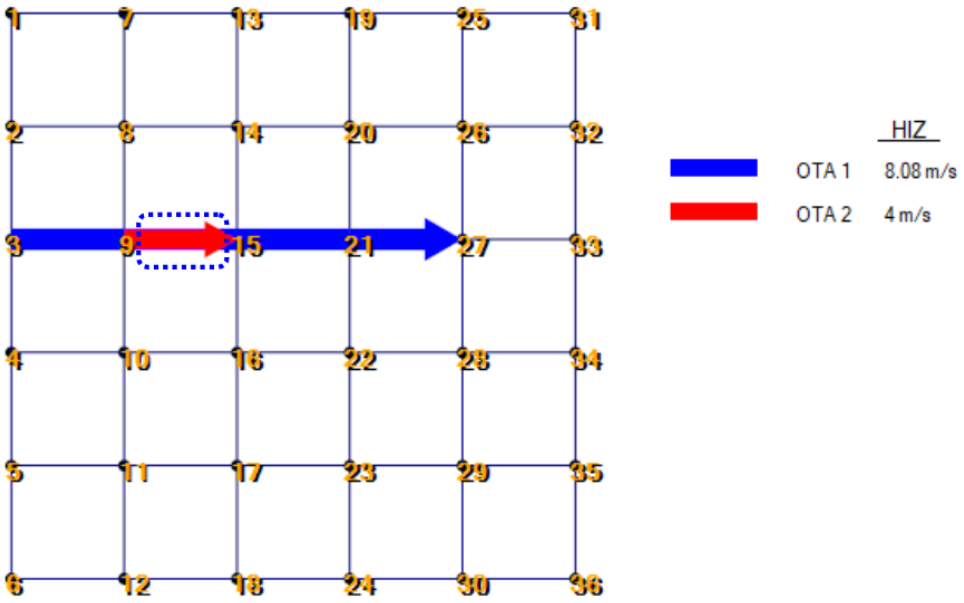
OTA	$DÜGÜM^{\substack{Düğüme\_gelis\_zamani(t_g)\\ Düğümden\_cikis\_zamani(t_c)}}$
OTA_1	$4^{(0.00)}_{(0.01)} \rightarrow 10^{(1.01)}_{(1.02)} \rightarrow 16^{(2.02)}_{(2.03)} \rightarrow 22^{(3.03)}_{(-)}$
OTA_2	$14^{(0.00)}_{(0.01)} \rightarrow 15^{(1.01)}_{(1.02)} \rightarrow 16^{(2.04)}_{(2.05)} \rightarrow 17^{(3.05)}_{(-)}$

Test-3 hız farkından kaynaklanan arka arkaya düğüm çakışması durumunu ele almaktadır. Bu test, rotası 3-27 olan OTA_1 ile rotası 9-15 olan OTA_2 için oluşturulmuştur. Test-4'ün ortam üzerinde gösterimi Şekil 4.3'te verilmiştir. Buna göre 15 numaralı düğümde çakışma meydana gelmektedir. Bu çakışma tipi Bölüm 3.2.2.3'de açıklanan durumlardan

ilkine karşılık gelmektedir. Bu test için OTA_1 'i bekletme maliyeti OTA_1 'in rotasını değiştirme maliyetinden daha düşüktür.

Çizelge 4.4. Test-2 için OTA_1 ve OTA_2 'nin Çakışmalı/Çakışmasız Rota Uzunluğu ve Rota Tamamlanma Süreleri

OTA	Çakışmanın Dikkate Alınmadığı Rota Uzunluğu	Çakışmanın Dikkate Alındığı Rota Uzunluğu	Çakışmanın Dikkate Alınmadığı Rota Süresi	Çakışmanın Dikkate Alındığı Rota süresi
OTA_1	12m	12m	3.03s	3.03s
OTA_2	12m	12m	3.03s	3.05s
Maksimum	*12m*	*12m*	*3.03s*	*3.05s*



Şekil 4.3. İki OTA İçin Hız Farkından Kaynaklanan Arka Arkaya Düğüm Çakışması

Önerilen algoritma çakışmayı önlemek için OTA_1 'i 9-15 kenarında (Şekil 4.3'te mavi kesik çizgili dikdörtgen ile gösterilmiştir) numaralı düğümde OTA_2 15 numaralı düğümü terk edene kadar bekletmiştir. Çizelge 4.5 tüm OTA'lar için önerilen algoritmanın bulduğu rotaları ve bu rotalar üzerindeki her düğüme giriş-çıkış zamanlarını göstermektedir. Çizelge 4.6 her OTA için çakışmaların dikkate alındığı ve alınmadığı rota uzunlukları ile rota sürelerini göstermektedir. Buna göre çakışmalar dikkate alınmadan hesaplanan toplam rota

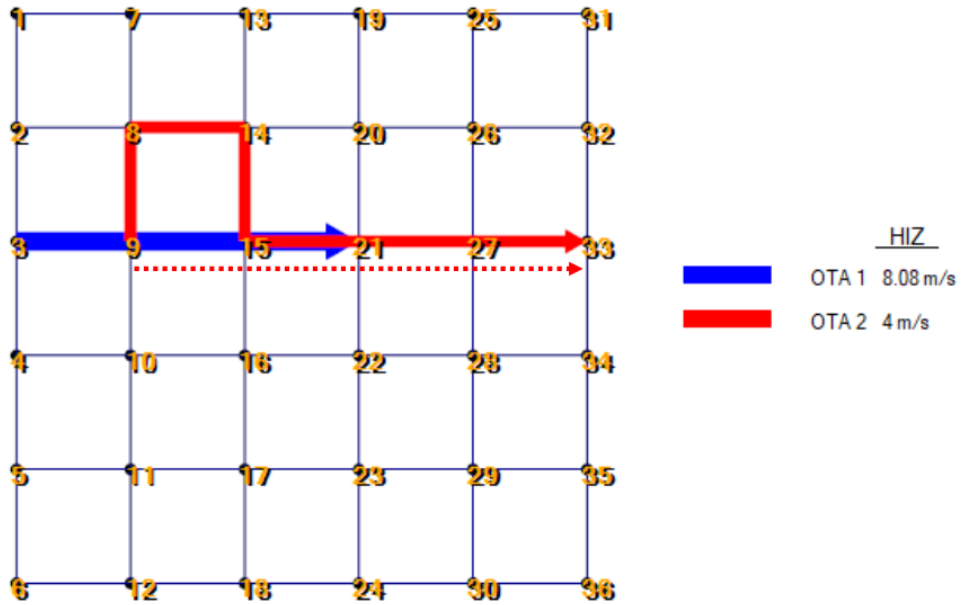
uzunluğu 20m, süre 3.030s iken, önerilen algoritmanın çakışmaları çözerek bulduğu rota uzunluğu 20m, süre 3.050s'dir.

Çizelge 4.5. Test-3 için OTA_1 ve OTA_2'nin Düğümlere Giriş Çıkış Zamanları

OTA	$DÜGÜM^{Düğüme\_gelis\_zamani(t_g)}_{Düğümden\_cikis\_zamani(t_c)}$
OTA_1	$3^{(0.000)}_{(0.010)} \rightarrow 9^{(0.505)}_{(0.515)} \rightarrow 15^{(1.030)}_{(1.040)} \rightarrow 21^{(1.535)}_{(1.545)} \rightarrow 27^{(2.040)}_{(-)}$
OTA_2	$9^{(0.00)}_{(0.01)} \rightarrow 15^{(1.01)}_{(-)}$

Çizelge 4.6. Test-3 için OTA_1 ve OTA_2'nin Çakışmalı/Çakışmasız Rota Uzunluğu ve Rota Tamamlanma Süreleri

OTA	**Çakışmanın Dikkate Alınmadığı Rota Uzunluğu**	**Çakışmanın Dikkate Alındığı Rota Uzunluğu**	**Çakışmanın Dikkate Alınmadığı Rota Süresi**	**Çakışmanın Dikkate Alındığı Rota süresi**
OTA_1	16m	16m	2.020s	2.040s
OTA_2	4m	4m	1.01s	1.01s
Maksimum	*16m*	*16m*	*2.020s*	*2.040s*



Şekil 4.4. İki OTA İçin Hız Farkından Kaynaklanan Arka Arkaya Düğüm Çakışması

Test-4 hız farkından kaynaklanan arka arkaya düğüm çakışması durumunu ele almaktadır. Bu test, rotası 3-21 olan OTA_1 ile rotası 9-33 olan OTA_2 için oluşturulmuştur. Test-4'ün ortam üzerinde gösterimi Şekil 4.4'te verilmiştir. Buna göre 15 numaralı düğümde çakışma meydana gelmektedir. Bu çakışma tipi Bölüm 3.2.2.1'de açıklanan durumlardan ikincisine karşılık gelmektedir. Önerilen algoritma bu duruma çözüm olarak çakışmayı önlemek için OTA_2 'nin rotasını değiştirmiştir. Şekil 4.4'te OTA_2 'nin çakışma olmaması durumunda gideceği rota kesik çizgili ok ile gösterilirken önerilen algoritmanın oluşturduğu rota düz okla gösterilmiştir.

Çizelge 4.7 tüm OTA'lar için önerilen algoritmanın bulduğu rotaları ve bu rotalar üzerindeki her düğüme giriş-çıkış zamanlarını göstermektedir. Çizelge 4.8 her OTA için çakışmaların dikkate alındığı ve alınmadığı rota uzunlukları ile rota sürelerini göstermektedir. Buna göre çakışmalar dikkate alınmadan hesaplanan toplam rota uzunluğu 28m, süre 5.555s iken, önerilen algoritmanın çakışmaları çözerek bulduğu rota uzunluğu 36m, süre 7.575s'dir.

Çizelge 4.7. Test-4 için OTA_1 ve OTA_2 'nin Düğümlere Giriş Çıkış Zamanları

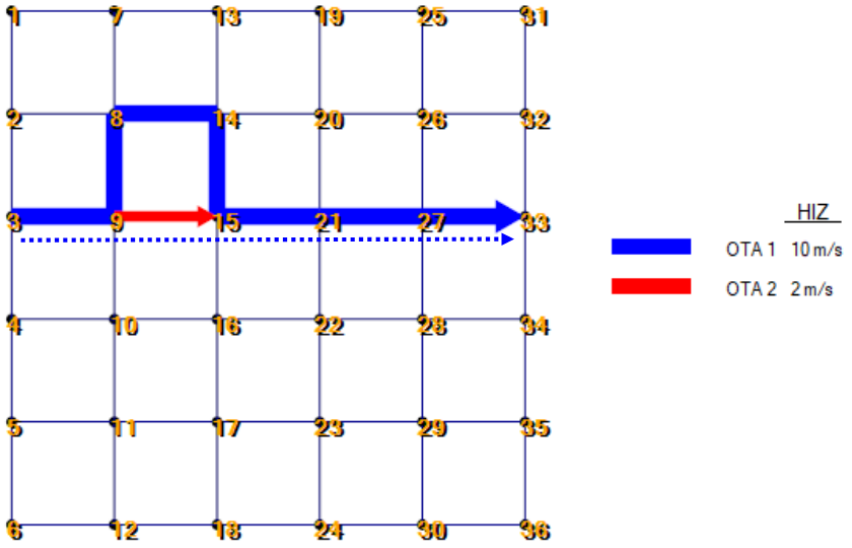
OTA	$DÜGÜM^{Düğüme\_gelis\_zamani(t_g)}_{Düğümden\_cikis\_zamani(t_c)}$
OTA_1	$3^{(0.000)}_{(0.010)} \rightarrow 9^{(0.505)}_{(0.515)} \rightarrow 15^{(1.010)}_{(1.020)} \rightarrow 21^{(1.515)}_{(-)}$
OTA_2	$9^{(0.00)}_{(0.01)} \rightarrow 8^{(1.01)}_{(1.02)} \rightarrow 14^{(2.02)}_{(2.03)} \rightarrow 15^{(3.03)}_{(3.04)} \rightarrow$ $21^{(4.04)}_{(4.05)} \rightarrow 27^{(5.05)}_{(5.06)} \rightarrow 33^{(6.06)}_{(-)}$

Çizelge 4.8. Test-4 için OTA_1 ve OTA_2 'nin Çakışmalı/Çakışmasız Rota Uzunluğu ve Rota Tamamlanma Süreleri

OTA	**Çakışmanın Dikkate Alınmadığı Rota Uzunluğu**	**Çakışmanın Dikkate Alındığı Rota Uzunluğu**	**Çakışmanın Dikkate Alınmadığı Rota Süresi**	**Çakışmanın Dikkate Alındığı Rota süresi**
OTA_1	12m	12m	1.515s	1.515s
OTA_2	16m	24m	4.04s	6.06s
Maksimum	*16m*	*24m*	*4.04s*	*6.06s*

Test-5 hız farkından kaynaklanan arka arkaya kenar çakışması durumunu ele almaktadır. Bu test, rotası 3-27 olan OTA_1 ile rotası 9-21 olan OTA_2 için oluşturulmuştur. Test-5'in ortam üzerinde gösterimi Şekil 4.6'da verilmiştir. Buna göre 9 ile 15 numaralı düğümler üzerindeki kenarda çakışma meydana gelmektedir. Bu çakışma tipi Bölüm 3.2.2.1'de açıklanan durumlardan ilkine karşılık gelmektedir. Bu test için OTA_1'in rotasını değiştirmenin maliyeti OTA_1'i bekletme maliyetinden daha düşüktür. Önerilen algoritma bu duruma çözüm olarak çakışmayı önlemek için OTA_1'in rotasını değiştirmiştir. Şekil 4.6'da OTA_1'in çakışma olmaması durumunda gideceği rota kesik çizgili ok ile gösterilirken önerilen algoritmanın oluşturduğu rota düz okla gösterilmiştir.

Çizelge 4.9 tüm OTA'lar için önerilen algoritmanın bulduğu rotaları ve bu rotalar üzerindeki her düğüme giriş-çıkış zamanlarını göstermektedir. Çizelge 4.10 her OTA için çakışmaların dikkate alındığı ve alınmadığı rota uzunlukları ile rota sürelerini göstermektedir. Buna göre çakışmalar dikkate alınmadan hesaplanan toplam rota uzunluğu 24m, süre 4.06s iken, önerilen algoritmanın çakışmaları çözerek bulduğu rota uzunluğu 32m, süre 4.88s'dir.



Şekil 4.5. İki OTA İçin Hız Farkından Kaynaklanan Arka Arkaya Kenar Çakışması

Çizelge 4.9. Test-5 için OTA_1 ve OTA_2 'nin Düğümlere Giriş Çıkış Zamanları

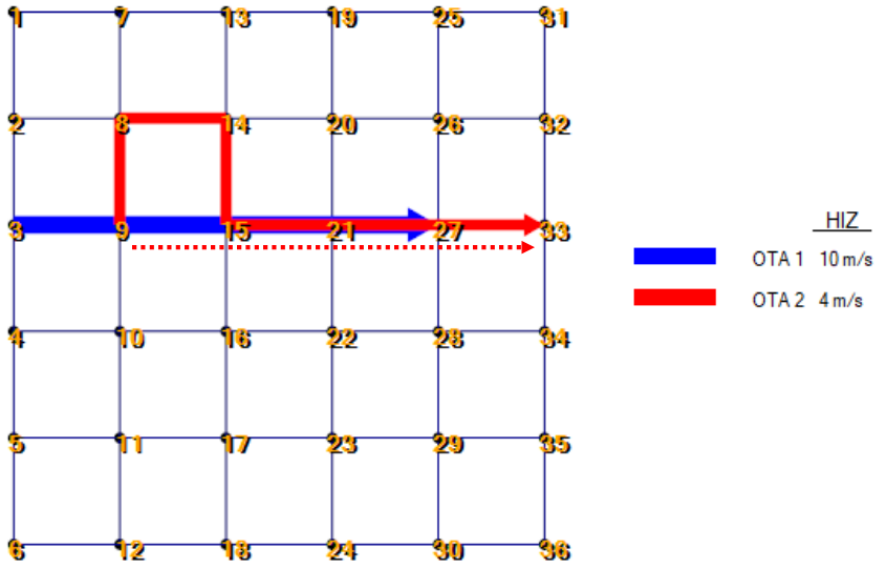
OTA	$DÜGÜM^{Dügüme\_gelis\_zamani(t_g)}_{Dügümden\_cikis\_zamani(t_c)}$
OTA_1	$3^{(0.00)}_{(0.01)} \rightarrow 9^{(0.41)}_{(0.42)} \rightarrow 8^{(0.82)}_{(0.83)} \rightarrow 14^{(1.23)}_{(1.24)} \rightarrow 15^{(1.64)}_{(1.65)} \rightarrow$ $21^{(2.05)}_{(2.06)} \rightarrow 27^{(2.46)}_{(2.47)} \rightarrow 33^{(2.87)}_{(-)}$
OTA_2	$9^{(0.00)}_{(0.01)} \rightarrow 15^{(2.01)}_{(-)}$

Çizelge 4.10. Test-5 için OTA_1 ve OTA_2 'nin Çakışmalı/Çakışmasız Rota Uzunluğu ve Rota Tamamlanma Süreleri

OTA	Çakışmanın Dikkate Alınmadığı Rota Uzunluğu	Çakışmanın Dikkate Alındığı Rota Uzunluğu	Çakışmanın Dikkate Alınmadığı Rota Süresi	Çakışmanın Dikkate Alındığı Rota süresi
OTA_1	20m	28m	2.05s	2.87s
OTA_2	4m	4m	2.01s	2.01s
Maksimum	*20m*	*28m*	*2.05s*	*2.87s*

Test-6 hız farkından kaynaklanan arka arkaya kenar çakışması durumunu ele almaktadır. Bu test, rotası 3-27 olan OTA_1 ile rotası 9-33 olan OTA_2 için oluşturulmuştur. Test-6'nın ortam üzerinde gösterimi Şekil 4.6'da verilmiştir. Buna göre 9 ile 15 numaralı düğümler üzerindeki kenarda çakışma meydana gelmektedir. Bu çakışma tipi Bölüm 3.2.2.1'de açıklanan durumlardan ikincisine karşılık gelmektedir. Önerilen algoritma bu duruma çözüm olarak çakışmayı önlemek için OTA_2 'nin rotasını değiştirmiştir. Şekil 4.6'da OTA_2 'nin çakışma olmaması durumunda gideceği rota kesik çizgili ok ile gösterilirken önerilen algoritmanın oluşturduğu rota düz okla gösterilmiştir.

Çizelge 4.11 tüm OTA'lar için önerilen algoritmanın bulduğu rotaları ve bu rotalar üzerindeki her düğüme giriş-çıkış zamanlarını göstermektedir. Çizelge 4.12 her OTA için çakışmaların dikkate alındığı ve alınmadığı rota uzunlukları ile rota sürelerini göstermektedir. Buna göre çakışmalar dikkate alınmadan hesaplanan toplam rota uzunluğu 32m, süre 5.68s iken, önerilen algoritmanın çakışmaları çözerek bulduğu rota uzunluğu 40m, süre 7.70s'dir.



Şekil 4.6. İki OTA İçin Hız Farkından Kaynaklanan Arka Arkaya Kenar Çakışması

Çizelge 4.11. Test-6 için OTA_1 ve OTA_2'nin Düğümlere Giriş Çıkış Zamanları

OTA	$DÜGÜM^{Dügüme\_gelis\_zamani(t_g)}_{Dügümden\_cikis\_zamani(t_c)}$
OTA_1	$3^{(0.00)}_{(0.01)} \rightarrow 9^{(0.41)}_{(0.42)} \rightarrow 15^{(0.82)}_{(0.83)} \rightarrow 21^{(1.23)}_{(1.24)} \rightarrow 27^{(1.64)}_{(-)}$
OTA_2	$9^{(0.00)}_{(0.01)} \rightarrow 8^{(1.01)}_{(1.02)} \rightarrow 14^{(2.02)}_{(2.03)} \rightarrow 15^{(3.03)}_{(3.04)} \rightarrow 21^{(4.04)}_{(4.05)} \rightarrow 27^{(5.05)}_{(5.06)} \rightarrow 33^{(6.06)}_{(-)}$

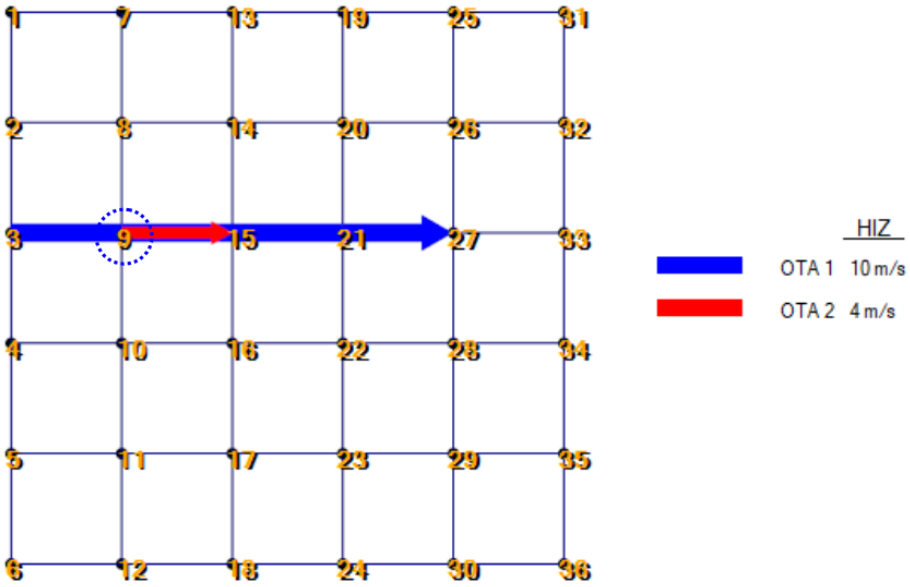
Çizelge 4.12. Test-6 için OTA_1 ve OTA_2'nin Çakışmalı/Çakışmasız Rota Uzunluğu ve Rota Tamamlanma Süreleri

OTA	**Çakışmanın Dikkate Alınmadığı Rota Uzunluğu**	**Çakışmanın Dikkate Alındığı Rota Uzunluğu**	**Çakışmanın Dikkate Alınmadığı Rota Süresi**	**Çakışmanın Dikkate Alındığı Rota süresi**
OTA_1	16m	16m	1.64s	1.64s
OTA_2	16m	24m	4.04s	6.06s
Maksimum	*16m*	*24m*	*4.04s*	*6.06s*

Test-7 hız farkından kaynaklanan arka arkaya kenar çakışması durumunu ele almaktadır. Bu örnek, rotası 3-27 olan OTA_1 ile rotası 9-15 olan OTA_2 için oluşturulmuştur. Test-7'nin ortam üzerinde gösterimi Şekil 4.7'de verilmiştir. Buna göre 9 ile 15 numaralı düğümler üzerindeki kenarda çakışma meydana gelmektedir. Bu çakışma tipi Bölüm 3.2.2.1'de açıklanan durumlardan ilkine karşılık gelmektedir. Bu test için OTA_1'i bekletme maliyeti OTA_1'in rotasını değiştirme maliyetinden daha düşüktür. Önerilen algoritma

çakışmayı önlemek için OTA_1 'i 9 (Şekil 4.7'de kesik çizgili daire ile gösterilmiştir) numaralı düğümde $T\_dügüm\_bekleme$ zamanı (OTA_2 15 numaralı düğümü terk edene kadar) bekletmiştir.

Çizelge 4.13 tüm OTA'lar için önerilen algoritmanın bulduğu rotaları ve bu rotalar üzerindeki her düğüme giriş-çıkış zamanlarını göstermektedir. Çizelge 4.14 her OTA için çakışmaların dikkate alındığı ve alınmadığı rota uzunlukları ile rota sürelerini göstermektedir. Buna göre çakışmalar dikkate alınmadan hesaplanan toplam rota uzunluğu 20m, süre 2.65s iken, önerilen algoritmanın çakışmaları çözerek bulduğu rota uzunluğu 20m, süre 2.85s'dir.



Şekil 4.7. İki OTA İçin Hız Farkından Kaynaklanan Arka Arkaya Kenar Çakışması

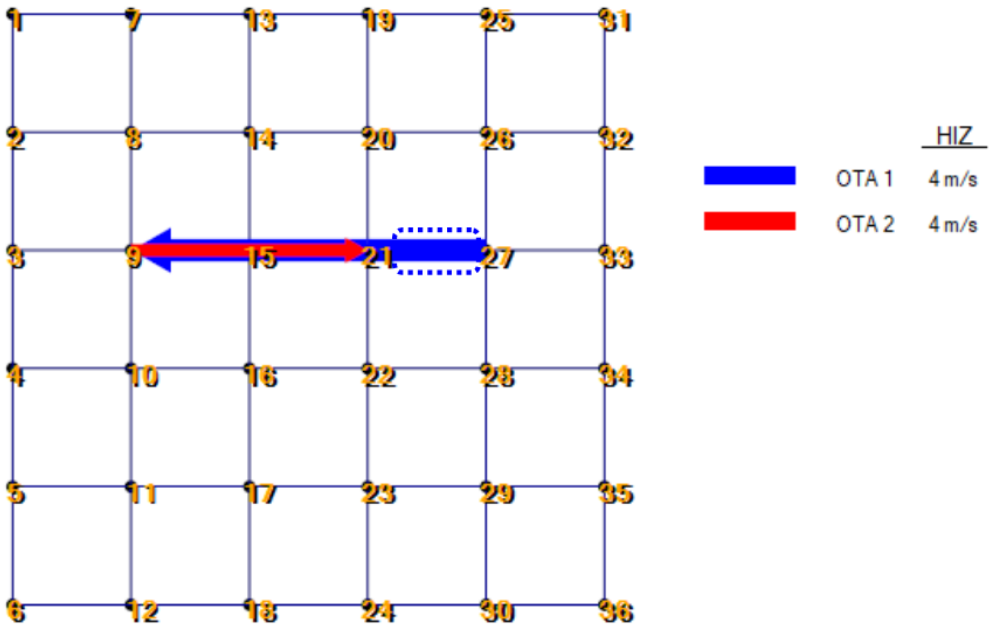
Çizelge 4.13. Test-7 için OTA_1 ve OTA_2 'nin Düğümlere Giriş Çıkış Zamanları

OTA	$DÜGÜM^{\substack{Dügüme\_gelis\_zamani(t_g)\\ Dügümden\_cikis\_zamani(t_c)}}$
OTA_1	$3^{(0.00)}_{(0.01)} \rightarrow 9^{(0.41)}_{(0.62)} \rightarrow 15^{(1.02)}_{(1.03)} \rightarrow 21^{(1.43)}_{(1.44)} \rightarrow 27^{(1.84)}_{(-)}$
OTA_2	$9^{(0.00)}_{(0.01)} \rightarrow 15^{(1.01)}_{(-)}$

Çizelge 4.14. Test-7 için OTA_1 ve OTA_2'nin Çakışmalı/Çakışmasız Rota Uzunluğu ve Rota Tamamlanma Süreleri

OTA	Çakışmanın Dikkate Alınmadığı Rota Uzunluğu	Çakışmanın Dikkate Alındığı Rota Uzunluğu	Çakışmanın Dikkate Alınmadığı Rota Süresi	Çakışmanın Dikkate Alındığı Rota süresi
OTA_1	16m	16m	1.64s	1.84s
OTA_2	4m	4m	1.01s	1.01s
Maksimum	*16m*	*16m*	*1.64s*	*1.84s*

Test-8 kenar üzerinde karşılıklı çakışma durumunu ele almaktadır. Bu test, rotası 27-9 olan OTA_1 ile rotası 9-27 olan OTA_2 için oluşturulmuştur. Test-8'in ortam üzerinde gösterimi Şekil 4.8'de verilmiştir. Buna göre 15 ile 21 numaralı düğümler arasındaki kenarda çakışma meydana gelmektedir. Bu çakışma tipi Bölüm 3.2.2.2'de açıklanan duruma karşılık gelmektedir. Önerilen algoritma çakışmayı önlemek için OTA_1'i 27-21 kenarında (Şekil 4.8'de mavi kesik çizgili dikdörtgen ile gösterilmiştir) OTA_2 21 numaralı düğümü terk edene kadar bekletmiştir.



Şekil 4.8. İki OTA İçin Kenar Üzerinde Karşılıklı Çakışma

Çizelge 4.15 tüm OTA'lar için önerilen algoritmanın bulduğu rotaları ve bu rotalar üzerindeki her düğüme giriş-çıkış zamanlarını göstermektedir. Çizelge 4.16 her OTA için çakışmaların dikkate alındığı ve alınmadığı rota uzunlukları ile rota sürelerini göstermektedir. Buna göre çakışmalar dikkate alınmadan hesaplanan toplam rota uzunluğu 20m, süre 5.05s iken, önerilen algoritmanın çakışmaları çözerek bulduğu rota uzunluğu 20m, süre 6.08s'dir.

Çizelge 4.15. Test-8 için OTA_1 ve OTA_2'nin Düğümlere Giriş Çıkış Zamanları

OTA	$DÜGÜM^{\substack{Dügüme\_gelis\_zamani(t_g)\\ Dügümden\_cikis\_zamani(t_c)}}$
OTA_1	$27^{(0.00)}_{(0.01)} \rightarrow 21^{(2.04)}_{(2.05)} \rightarrow 15^{(3.05)}_{(3.06)} \rightarrow 9^{(4.06)}_{(-)}$
OTA_2	$9^{(0.00)}_{(0.01)} \rightarrow 15^{(1.01)}_{(1.02)} \rightarrow 21^{(2.02)}_{(-)}$

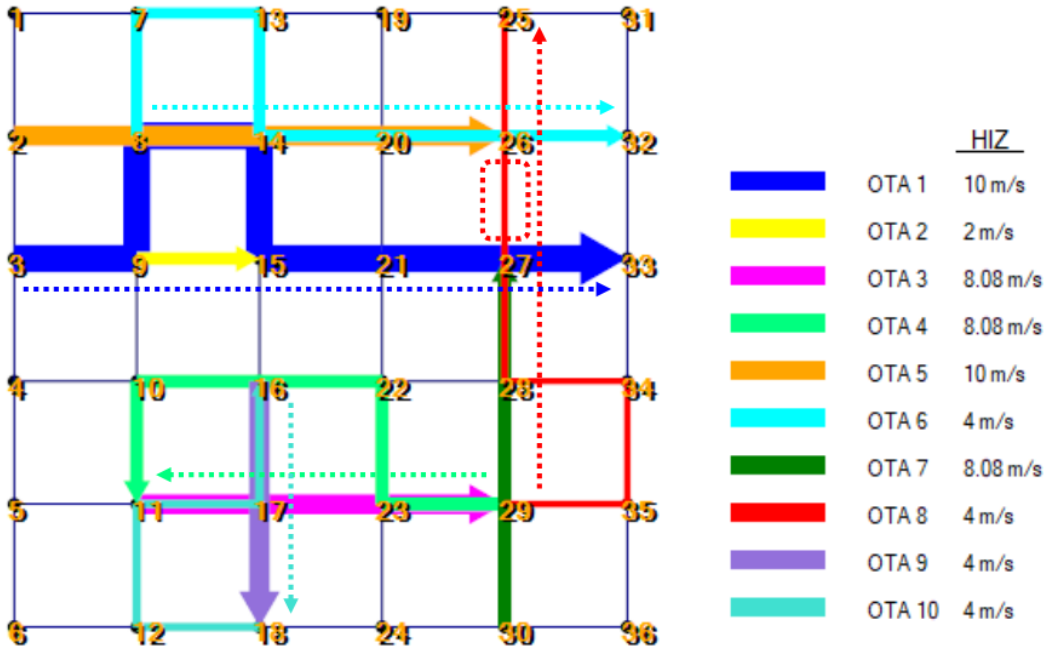
Çizelge 4.16. Test-8 için OTA_1 ve OTA_2'nin Çakışmalı/Çakışmasız Rota Uzunluğu ve Rota Tamamlanma Süreleri

OTA	**Çakışmanın Dikkate Alınmadığı Rota Uzunluğu**	**Çakışmanın Dikkate Alındığı Rota Uzunluğu**	**Çakışmanın Dikkate Alınmadığı Rota Süresi**	**Çakışmanın Dikkate Alındığı Rota süresi**
OTA_1	12m	12m	3.03s	4.06s
OTA_2	8m	8m	2.02s	2.02s
Maksimum	*12m*	*12m*	*3.03s*	*4.06s*

Yukarıda açıklanan sekiz adet test iki OTA arasında meydana gelebilecek çakışma tiplerini ve önerilen algoritmanın çözümlerini ayrıntılı olarak ifade etmektedir. Yapılan testlerde çakışmayı önlemek için oluşturulan rotaların uzunluğunun ve süresinin, çakışmanın dikkate alınmadığı rotalara göre arttığı görülmektedir. Ancak çakışma durumu dikkate alınmadığı takdirde OTA'lardan biri diğerinin rotasını tamamlamasını bekleyecekti ya da her iki OTA karşılıklı olarak hareket ettiği durumlarda OTA'lar birbirlerini bekleyeceklerdi. Örneğin Test-2'de çakışma durumu önceden düşünülmediği takdirde 16 numaralı düğümde meydana gelen çakışma için OTA_2 OTA_1'in görevini bitirmesini bekleyecek ve rotasını 4.04 s'de tamamlayacaktı. OTA_2'nin önerilen algoritmayla hesaplanan çakışmasız rotası ise 3.05 s sürmektedir. Bu durumda çakışma durumu hesaba katılarak hesaplanan rota daha avantajlı olmaktadır.

4.2. On OTA için Oluşturulan Testler

Şekil 4.9'daki Test-9 aynı ortamda hareket eden heterojen on OTA için oluşturulmuştur. Ortamdaki tüm OTA'ların başlangıç ve bitiş düğümleri Çizelge 4.18'de verilmiştir. Bu testte beş adet çakışma tipine ait toplam altı çakışma meydana gelmiştir. Önerilen algoritma bu çakışma tiplerinin tamamını aynı test içerisinde çözebilmiştir. OTA_1 ile OTA_2 arasında 9-15 kenarı üzerinde hız farkından kaynaklanan arka arkaya kenar çakışması durumu oluşmaktadır. Önerilen algoritma burada OTA_1 'in rotasını değişirmiştir. OTA_5 ile OTA_6 arasında 8-14 kenarı üzerinde hız farkından kaynaklanan arka arkaya kenar çakışması durumu oluşmaktadır. Önerilen algoritma burada OTA_6 'nın rotasını değişirmiştir. OTA_7 ile OTA_8 arasında 28 numaralı düğümde hız farkından kaynaklanan arka arkaya düğüm çakışması durumu oluşmaktadır. Önerilen algoritma burada OTA_8 'in rotasını değişirmiştir. OTA_3 ile OTA_4 arasında 17-23 kenarı üzerinde kenar üzerinde karşılıklı çakışma durumu oluşmaktadır. Önerilen algoritma burada OTA_4 'ün rotasını değişirmiştir. OTA_9 ile OTA_{10} arasında 17 numaralı düğümde aynı doğrultuda düğüm üzerinde karşılıklı çakışma durumu oluşmaktadır.



Şekil 4.9. Heterojen On OTA için Yapılan Test-9'un Çakışmalı/Çakışmasız Rota Gösterimi

Çizelge 4.17. Test-9 için Heterojen On OTA'nın Düğümlere Giriş Çıkış Zamanları

OTA	$DÜGÜM^{Düğüme\_gelis\_zamani(t_g)}_{Düğümden\_cikis\_zamani(t_c)}$
OTA_1	$3^{(0.00)}_{(0.01)} \rightarrow 9^{(0.41)}_{(0.42)} \rightarrow 8^{(0.82)}_{(0.83)} \rightarrow 14^{(1.23)}_{(1.24)} \rightarrow 15^{(1.64)}_{(1.65)} \rightarrow 21^{(2.05)}_{(2.06)} \rightarrow 27^{(2.46)}_{(2.47)} \rightarrow 33^{(2.87)}_{(-)}$
OTA_2	$9^{(0.00)}_{(0.01)} \rightarrow 15^{(2.01)}_{(-)}$
OTA_3	$11^{(0.00)}_{(0.01)} \rightarrow 17^{(0.505)}_{(0.515)} \rightarrow 23^{(1.010)}_{(1.020)} \rightarrow 29^{(1.51)}_{(-)}$
OTA_4	$29^{(0.00)}_{(0.01)} \rightarrow 23^{(0.505)}_{(0.515)} \rightarrow 22^{(1.010)}_{(1.020)} \rightarrow 16^{(1.515)}_{(1.525)} \rightarrow 10^{(2.020)}_{(2.030)} \rightarrow 11^{(2.525)}_{(-)}$
OTA_5	$2^{(0.00)}_{(0.01)} \rightarrow 8^{(0.41)}_{(0.42)} \rightarrow 14^{(0.82)}_{(0.83)} \rightarrow 20^{(1.23)}_{(1.24)} \rightarrow 26^{(1.64)}_{(-)}$
OTA_6	$8^{(0.00)}_{(0.01)} \rightarrow 7^{(1.01)}_{(1.02)} \rightarrow 13^{(2.02)}_{(2.03)} \rightarrow 14^{(3.03)}_{(3.04)} \rightarrow 20^{(4.04)}_{(4.05)} \rightarrow 26^{(5.05)}_{(5.06)} \rightarrow 32^{(6.06)}_{(-)}$
OTA_7	$30^{(0.00)}_{(0.01)} \rightarrow 29^{(0.505)}_{(0.515)} \rightarrow 28^{(1.010)}_{(1.020)} \rightarrow 27^{(1.515)}_{(-)}$
OTA_8	$29^{(0.00)}_{(0.01)} \rightarrow 35^{(1.01)}_{(1.02)} \rightarrow 34^{(2.02)}_{(2.03)} \rightarrow 28^{(3.03)}_{(3.04)} \rightarrow 27^{(4.04)}_{(4.05)} \rightarrow 26^{(5.07)}_{(5.08)} \rightarrow 25^{(6.08)}_{(-)}$
OTA_9	$16^{(0.00)}_{(0.01)} \rightarrow 17^{(1.01)}_{(1.02)} \rightarrow 18^{(2.02)}_{(-)}$
OTA_{10}	$18^{(0.00)}_{(0.01)} \rightarrow 12^{(1.01)}_{(1.02)} \rightarrow 11^{(2.02)}_{(2.03)} \rightarrow 17^{(3.03)}_{(3.04)} \rightarrow 16^{(4.04)}_{(-)}$

Önerilen algoritma burada OTA_{10} 'un rotasını değişirmiştir. Şekil 4.9'da OTA_1, OTA_4, OTA_6, OTA_8 ve OTA_{10} 'un çakışma olmaması durumunda gidecekleri rotalar kesik çizgili ok (sırasıyla lacivert, açık yeşil, turkuaz, kırmızı ve mavi) ile gösterilirken önerilen algoritmanın oluşturduğu rotalar düz okla (sırasıyla lacivert, açık yeşil, turkuaz, kırmızı ve mavi) gösterilmiştir. OTA_6 ile OTA_8 arasında 26 numaralı düğümde farklı doğrultuda düğüm üzerinde karşılıklı çakışma durumu oluşmaktadır. Önerilen algoritma çakışmayı önlemek için OTA_8 'i 27-26 kenarında (Şekil 4.9'da kesikli kırmızı çizgili dikdörtgen ile gösterilmiştir) OTA_6 26 numaralı düğümü terk edene kadar bekletmiştir.

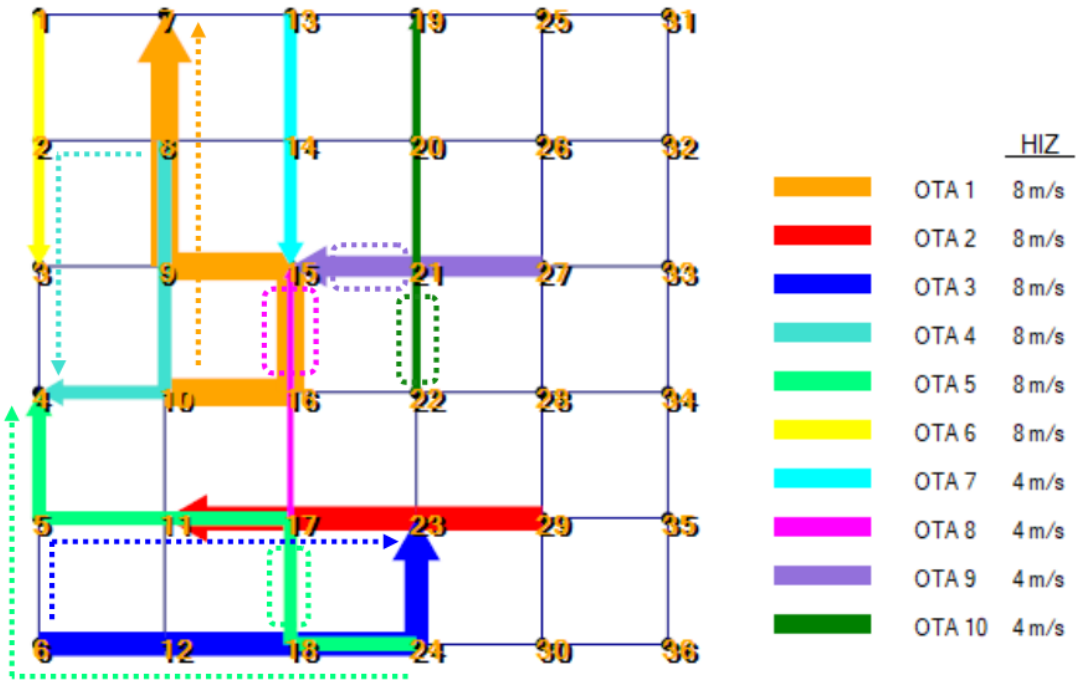
Çizelge 4.17 tüm OTA'lar için önerilen algoritmanın bulduğu rotaları ve bu rotalar üzerindeki her düğüme giriş-çıkış zamanlarını göstermektedir. Çizelge 4.18 her OTA için çakışmaların dikkate alındığı ve alınmadığı rota uzunlukları ile rota sürelerini göstermektedir. Buna göre çakışmalar dikkate alınmadan hesaplanan toplam rota uzunluğu 124m, süre 22.365s iken, önerilen algoritmanın çakışmaları çözerek bulduğu rota uzunluğu 164m, süre 30.275s'dir.

Çizelge 4.18. Test-9 için Heterojen On OTA'nın Çakışmalı/Çakışmasız Rota Uzunluğu ve Rota Tamamlanma Süreleri

OTA	Çakışmanın Dikkate Alınmadığı Rota Uzunluğu	Çakışmanın Dikkate Alındığı Rota Uzunluğu	Çakışmanın Dikkate Alınmadığı Rota Süresi	Çakışmanın Dikkate Alındığı Rota süresi
OTA_1	20m	28m	2.05s	2.87s
OTA_2	4m	4m	2.01s	2.01s
OTA_3	12m	12m	1.515s	1.515s
OTA_4	12m	20m	1.515s	2.525s
OTA_5	16m	16m	1.64s	1.64s
OTA_6	16m	24m	4.04s	6.06s
OTA_7	12m	12m	1.515s	1.515s
OTA_8	16m	24m	4.04s	6.08s
OTA_9	8m	8m	2.02s	2.02s
OTA_{10}	8m	16m	2.02s	4.04s
Maksimum	*20m*	*28m*	*4.04s*	*6.08s*

Şekil 4.10'da görülen Test-10 aynı ortamda hareket eden heterojen on OTA için oluşturulmuştur. Ortamdaki tüm OTA'ların başlangıç ve bitiş düğümleri Çizelge 4.20'de verilmiştir. Bu testte üç adet çakışma tipine ait toplam sekiz çakışma meydana gelmiştir. Önerilen algoritma bu çakışma tiplerinin tamamını aynı test içerisinde çözebilmiştir. OTA_4 ile OTA_6 arasında 2 numaralı düğüm üzerinde farklı doğrultuda düğüm üzerinde karşılıklı çakışma durumu meydana gelmektedir. Önerilen algoritma burada OTA_6 'ün rotasını değişirmiştir. OTA_1 ile OTA_4 arasında 9 numaralı düğüm üzerinde aynı doğrultuda düğüm üzerinde karşılıklı çakışma durumu meydana gelmektedir. Önerilen algoritma burada OTA_1 'in rotasını değişirmiştir. OTA_2 ile OTA_3 arasında 11-17 kenarı üzerinde kenar üzerinde karşılıklı çakışma durumu oluşmaktadır. Önerilen algoritma burada OTA_3 'ün rotasını değişirmiştir. OTA_3 ile OTA_5 arasında 12-18 kenarı üzerinde kenar üzerinde karşılıklı çakışma durumu oluşmaktadır. Önerilen algoritma burada OTA_5 'in rotasını değişirmiştir. OTA_7 ile OTA_8 arasında 15 numaralı düğümde aynı doğrultuda düğüm üzerinde karşılıklı çakışma durumu oluşmaktadır. Önerilen algoritma burada OTA_8 'in

rotasını değişirmiştir. Şekil 4.10'da OTA_1 , OTA_3 , OTA_4 , OTA_5 , OTA_8 'in çakışma olmaması durumunda gidecekleri rotalar kesik çizgili ok (sırasıyla turuncu, lacivert, mavi, açık yeşil ve pembe) ile gösterilirken önerilen algoritmanın oluşturduğu rotalar düz okla (sırasıyla sırasıyla turuncu, lacivert, mavi, açık yeşil ve pembe) gösterilmiştir.



Şekil 4.10. Heterojen On OTA için Yapılan Test-10'un Çakışmalı/Çakışmasız Rota Gösterimi

OTA_7 ile OTA_9 arasında 15 numaralı düğümde farklı doğrultuda düğüm üzerinde karşılıklı çakışma durumu oluşmaktadır. Önerilen algoritma çakışmayı önlemek için OTA_9 'u 21-15 kenarında (Şekil 4.10'da kesikli mor çizgili dikdörtgen ile gösterilmiştir) OTA_7 15 numaralı düğümü terk edene kadar bekletmiştir. OTA_9 ile OTA_{10} arasında 21 numaralı düğümde farklı doğrultuda düğüm üzerinde karşılıklı çakışma durumu oluşmaktadır.

Önerilen algoritma çakışmayı önlemek için OTA_{10} 'u 22-21 kenarında (Şekil 4.10'da kesikli yeşil çizgili dikdörgen ile gösterilmiştir) OTA_9 21 numaralı düğümü terk edene kadar bekletmiştir. OTA_3 ile OTA_5 arasında 17 numaralı düğümde farklı doğrultuda düğüm üzerinde karşılıklı çakışma durumu oluşmaktadır. Önerilen algoritma çakışmayı önlemek

için OTA_5 'i 18-17 kenarında (Şekil 4.10'da kesikli açık yeşil çizgili dikdörtgen ile gösterilmiştir) OTA_3 17 numaralı düğümü terk edene kadar bekletmiştir.

Çizelge 4.19. Test-10 için Heterojen On OTA'nın Düğümlere Giriş Çıkış Zamanları

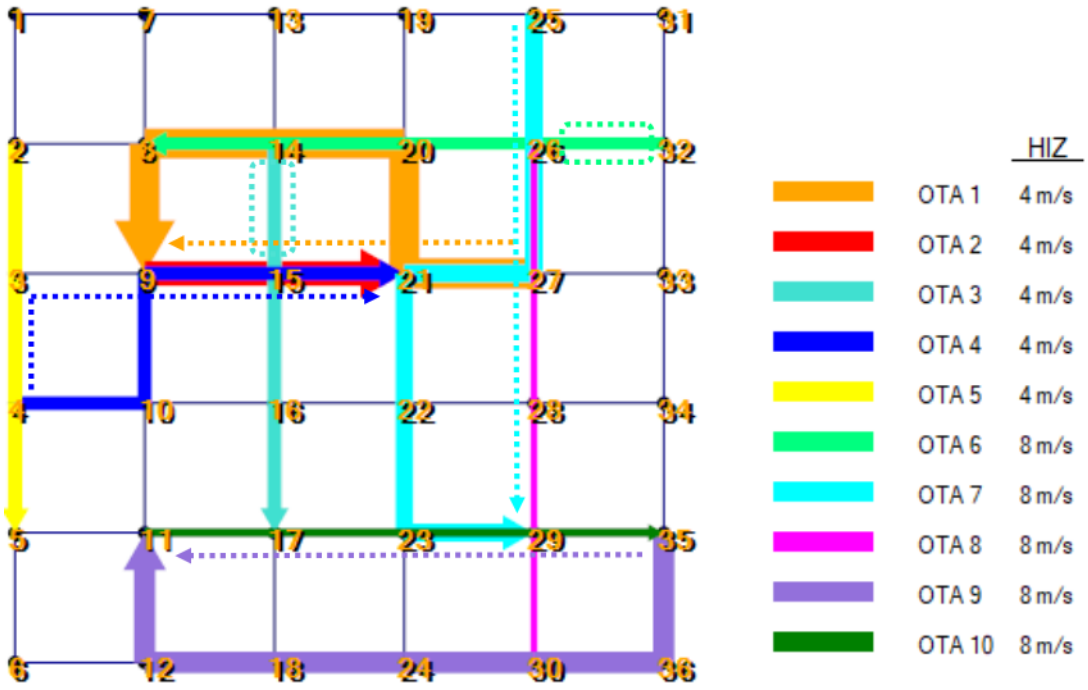
OTA	$DÜGÜM^{Düğüme\_gelis\_zamani(t_g)}_{Düğümden\_cikis\_zamani(t_c)}$
OTA_1	$10^{(0.00)}_{(0.01)} \rightarrow 16^{(0.51)}_{(0.52)} \rightarrow 15^{(1.02)}_{(1.03)} \rightarrow 9^{(1.53)}_{(1.54)} \rightarrow 8^{(2.04)}_{(2.05)} \rightarrow 7^{(2.55)}_{(-)}$
OTA_2	$29^{(0.00)}_{(0.01)} \rightarrow 23^{(0.51)}_{(0.52)} \rightarrow 17^{(1.02)}_{(1.03)} \rightarrow 11^{(1.53)}_{(-)}$
OTA_3	$6^{(0.00)}_{(0.01)} \rightarrow 12^{(0.51)}_{(0.52)} \rightarrow 18^{(1.02)}_{(1.03)} \rightarrow 24^{(1.53)}_{(1.54)} \rightarrow 23^{(2.04)}_{(-)}$
OTA_4	$8^{(0.00)}_{(0.01)} \rightarrow 9^{(0.51)}_{(0.52)} \rightarrow 10^{(1.02)}_{(1.03)} \rightarrow 4^{(1.53)}_{(-)}$
OTA_5	$24^{(0.00)}_{(0.01)} \rightarrow 18^{(0.51)}_{(0.52)} \rightarrow 17^{(1.04)}_{(1.05)} \rightarrow 11^{(1.55)}_{(1.56)} \rightarrow 5^{(2.06)}_{(2.07)} \rightarrow 4^{(2.57)}_{(-)}$
OTA_6	$1^{(0.00)}_{(0.01)} \rightarrow 2^{(0.51)}_{(0.52)} \rightarrow 3^{(1.02)}_{(-)}$
OTA_7	$13^{(0.00)}_{(0.01)} \rightarrow 14^{(1.01)}_{(1.02)} \rightarrow 15^{(2.02)}_{(-)}$
OTA_8	$17^{(0.00)}_{(0.01)} \rightarrow 16^{(1.01)}_{(1.02)} \rightarrow 15^{(2.04)}_{(-)}$
OTA_9	$27^{(0.00)}_{(0.01)} \rightarrow 21^{(1.01)}_{(1.02)} \rightarrow 15^{(2.06)}_{(-)}$
OTA_{10}	$22^{(0.00)}_{(0.01)} \rightarrow 21^{(1.03)}_{(1.04)} \rightarrow 20^{(2.04)}_{(2.05)} \rightarrow 19^{(3.05)}_{(-)}$

Çizelge 4.20. Test-10 için Heterojen On OTA'nın Çakışmalı/Çakışmasız Rota Uzunluğu ve Rota Tamamlanma Süreleri

OTA	**Çakışmanın Dikkate Alınmadığı Rota Uzunluğu**	**Çakışmanın Dikkate Alındığı Rota Uzunluğu**	**Çakışmanın Dikkate Alınmadığı Rota Süresi**	**Çakışmanın Dikkate Alındığı Rota süresi**
OTA_1	12m	20m	1.53s	2.55s
OTA_2	12m	12m	1.53s	1.53s
OTA_3	16m	16m	2.04s	2.04s
OTA_4	12m	12m	1.53s	1.53s
OTA_5	20m	20m	2.55s	2.57s
OTA_6	8m	8m	1.02s	1.02s
OTA_7	8m	8m	2.02s	2.02s
OTA_8	8m	8m	2.02s	2.04s
OTA_9	8m	8m	2.02s	2.06s
OTA_{10}	12m	12m	3.03s	3.05s
Maksimum	*20m*	*20m*	*3.03s*	*3.05s*

Çizelge 4.19 tüm OTA'lar için önerilen algoritmanın bulduğu rotaları ve bu rotalar üzerindeki her düğüme giriş-çıkış zamanlarını göstermektedir. Çizelge 4.20 her OTA için çakışmaların dikkate alındığı ve alınmadığı rota uzunlukları ile rota sürelerini göstermektedir. Buna göre çakışmalar dikkate alınmadan hesaplanan toplam rota uzunluğu 116m, süre 19.29s iken, önerilen algoritmanın çakışmaları çözerek bulduğu rota uzunluğu 124m, süre 20.41s'dir.

Şekil 4.11'de görülen Test-11 aynı ortamda hareket eden heterojen on OTA için oluşturulmuştur. Ortamdaki tüm OTA'ların başlangıç ve bitiş düğümleri Çizelge 4.22'de verilmiştir. Bu testte üç adet çakışma tipine ait toplam altı çakışma meydana gelmiştir. Önerilen algoritma bu çakışma tiplerinin tamamını aynı test içerisinde çözebilmiştir. OTA_1 ile OTA_2 arasında 11-17 kenarı üzerinde kenar üzerinde karşılıklı çakışma durumu oluşmaktadır. Önerilen algoritma burada OTA_1 'in rotasını değişirmiştir. OTA_4 ile OTA_5 arasında 3 numaralı düğüm üzerinde aynı doğrultuda düğüm üzerinde karşılıklı çakışma durumu meydana gelmektedir. Önerilen algoritma burada OTA_4 'ün rotasını değişirmiştir. OTA_7 ile OTA_8 arasında 27-28 kenarı üzerinde kenar üzerinde karşılıklı çakışma durumu oluşmaktadır. Önerilen algoritma burada OTA_7 'nin rotasını değişirmiştir. OTA_9 ile OTA_{10} arasında 23 numaralı düğümde aynı doğrultuda düğüm üzerinde karşılıklı çakışma durumu oluşmaktadır. Önerilen algoritma burada OTA_9 'un rotasını değişirmiştir. Şekil 4.11'de $OTA_1, OTA_4, OTA_7, OTA_9$ 'un çakışma olmaması durumunda gidecekleri rotalar kesik çizgili ok (sırasıyla turuncu, lacivert, turkuaz ve mor) ile gösterilirken önerilen algoritmanın oluşturduğu rotalar düz okla (sırasıyla turuncu, lacivert, turkuaz ve mor) gösterilmiştir. OTA_2 ile OTA_3 arasında 15 numaralı düğümde farklı doğrultuda düğüm üzerinde karşılıklı çakışma durumu oluşmaktadır. Önerilen algoritma çakışmayı önlemek için OTA_3 'ü 14-15 kenarında (Şekil 4.11'de kesikli mavi çizgili dikdörtgen ile gösterilmiştir) OTA_2 15 numaralı düğümü terk edene kadar bekletmiştir. OTA_6 ile OTA_7 arasında 26 numaralı düğümde farklı doğrultuda düğüm üzerinde karşılıklı çakışma durumu oluşmaktadır. Önerilen algoritma çakışmayı önlemek için OTA_6 'yı 26-32 kenarında (Şekil 4.11'de kesikli açık yeşil çizgili dikdörtgen ile gösterilmiştir) OTA_7 26 numaralı düğümü terk edene kadar bekletmiştir.



Şekil 4.11. Heterojen On OTA için Yapılan Test-11'in Çakışmalı/Çakışmasız Rota Gösterimi

Çizelge 4.21. Test-11 için Heterojen On OTA'nın Düğümlere Giriş Çıkış Zamanları

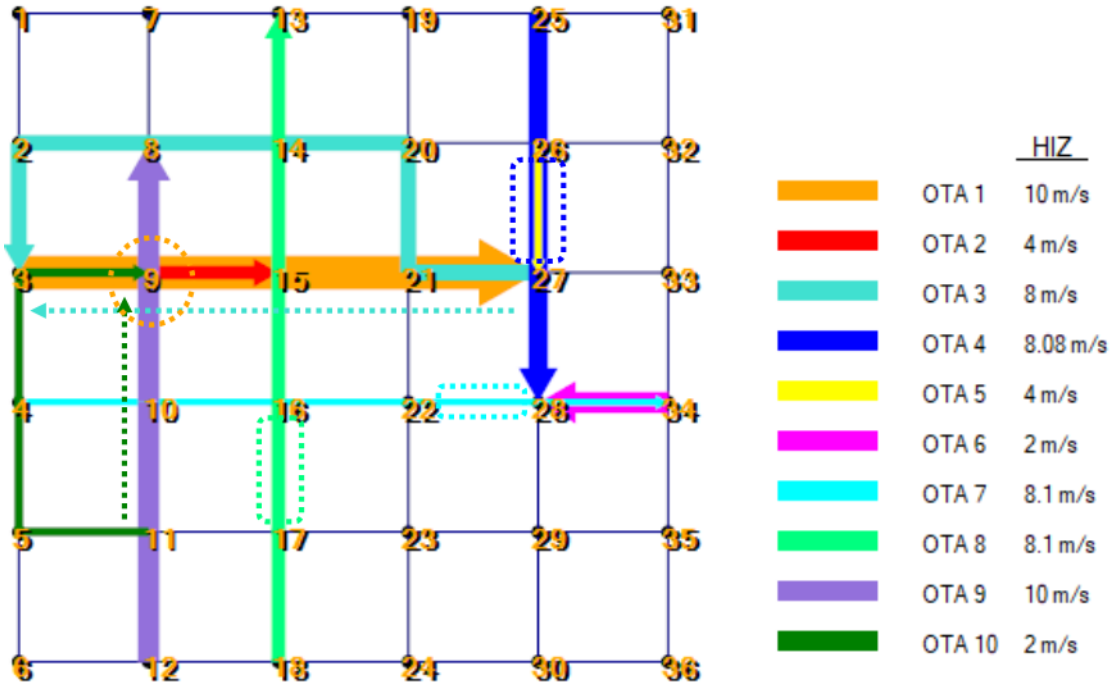
OTA	$DÜGÜM^{Düğüme\_gelis\_zamani(t_g)}_{Düğümden\_cikis\_zamani(t_c)}$
OTA_1	$27^{(0.00)}_{(0.01)} \rightarrow 21^{(1.01)}_{(1.02)} \rightarrow 20^{(2.02)}_{(2.03)} \rightarrow 14^{(3.03)}_{(3.04)} \rightarrow 8^{(4.04)}_{(4.05)} \rightarrow 9^{(5.05)}_{(-)}$
OTA_2	$9^{(0.00)}_{(0.01)} \rightarrow 15^{(1.01)}_{(1.02)} \rightarrow 21^{(2.02)}_{(-)}$
OTA_3	$14^{(0.00)}_{(0.01)} \rightarrow 15^{(1.03)}_{(1.04)} \rightarrow 16^{(2.04)}_{(2.05)} \rightarrow 17^{(3.05)}_{(-)}$
OTA_4	$4^{(0.00)}_{(0.01)} \rightarrow 10^{(1.01)}_{(1.02)} \rightarrow 9^{(2.02)}_{(2.03)} \rightarrow 15^{(3.03)}_{(3.04)} \rightarrow 21^{(4.04)}_{(-)}$
OTA_5	$2^{(0.00)}_{(0.01)} \rightarrow 3^{(1.01)}_{(1.02)} \rightarrow 4^{(2.02)}_{(2.03)} \rightarrow 5^{(3.03)}_{(-)}$
OTA_6	$32^{(0.00)}_{(0.01)} \rightarrow 26^{(0.53)}_{(0.54)} \rightarrow 20^{(1.04)}_{(1.05)} \rightarrow 14^{(1.55)}_{(1.56)} \rightarrow 8^{(2.06)}_{(-)}$
OTA_7	$25^{(0.00)}_{(0.01)} \rightarrow 26^{(0.51)}_{(0.52)} \rightarrow 27^{(1.02)}_{(1.03)} \rightarrow 21^{(1.53)}_{(1.54)} \rightarrow 22^{(2.04)}_{(2.05)} \rightarrow 23^{(2.55)}_{(2.56)} \rightarrow 29^{(3.06)}_{(-)}$
OTA_8	$30^{(0.00)}_{(0.01)} \rightarrow 29^{(0.51)}_{(0.52)} \rightarrow 28^{(1.02)}_{(1.03)} \rightarrow 27^{(1.53)}_{(1.54)} \rightarrow 26^{(2.04)}_{(-)}$
OTA_9	$35^{(0.00)}_{(0.01)} \rightarrow 36^{(0.51)}_{(0.52)} \rightarrow 30^{(1.02)}_{(1.03)} \rightarrow 24^{(1.53)}_{(1.54)} \rightarrow 18^{(2.04)}_{(2.05)} \rightarrow 12^{(2.55)}_{(2.56)} \rightarrow 11^{(3.06)}_{(-)}$
OTA_{10}	$11^{(0.00)}_{(0.01)} \rightarrow 17^{(0.51)}_{(0.52)} \rightarrow 23^{(1.02)}_{(1.03)} \rightarrow 29^{(1.53)}_{(1.54)} \rightarrow 35^{(2.04)}_{(-)}$

Çizelge 4.21 tüm OTA'lar için önerilen algoritmanın bulduğu rotaları ve bu rotalar üzerindeki her düğüme giriş-çıkış zamanlarını göstermektedir. Çizelge 4.22 her OTA için çakışmaların dikkate alındığı ve alınmadığı rota uzunlukları ile rota sürelerini göstermektedir. Buna göre çakışmalar dikkate alınmadan hesaplanan toplam rota uzunluğu 140m, süre 25.35s iken, önerilen algoritmanın çakışmaları çözerek bulduğu rota uzunluğu 164m, süre 29.45s'dir.

Çizelge 4.22. Test-11 için Heterojen On OTA'nın Çakışmalı/Çakışmasız Rota Uzunluğu ve Rota Tamamlanma Süreleri

OTA	Çakışmanın Dikkate Alınmadığı Rota Uzunluğu	Çakışmanın Dikkate Alındığı Rota Uzunluğu	Çakışmanın Dikkate Alınmadığı Rota Süresi	Çakışmanın Dikkate Alındığı Rota süresi
OTA_1	12m	20m	3.03s	5.05s
OTA_2	8m	8m	2.02s	2.02s
OTA_3	12m	12m	3.03s	3.05s
OTA_4	16m	16m	4.04s	4.04s
OTA_5	12m	12m	3.03s	3.03s
OTA_6	16m	16m	2.04s	2.06s
OTA_7	16m	24m	2.04s	3.06s
OTA_8	16m	16m	2.04s	2.04s
OTA_9	16m	24m	2.04s	3.06s
OTA_{10}	16m	16m	2.04s	2.04s
Maksimum	*16m*	*24m*	*4.04s*	*5.05s*

Şekil 4.12'de görülen Test-12 aynı ortamda hareket eden heterojen on OTA için oluşturulmuştur. Ortamdaki tüm OTA'ların başlangıç ve bitiş düğümleri Çizelge 4.24'de verilmiştir. Bu testte beş adet çakışma tipine ait toplam altı çakışma meydana gelmiştir. Önerilen algoritma bu çakışma tiplerinin tamamını aynı test içerisinde çözebilmiştir. OTA_4 ve OTA_5 arasında 27 numaralı düğümde hız farkından kaynaklanan arka arkaya düğüm çakışması durumu meydana gelmiştir. Önerilen algoritma çakışmayı önlemek için OTA_4'ü 26-27 kenarında (Şekil 4.12'de kesikli lacivert çizgili dikdörtgen ile gösterilmiştir) OTA_5 27 numaralı düğümü terk edene kadar bekletmiştir. OTA_6 ve OTA_7 arasında 28 numaralı düğümde aynı doğrultuda düğüm üzerinde karşılıklı çakışma durumu meydana gelmektedir.



Şekil 4.12. Heterojen On OTA için Yapılan Test-12'nin Çakışmalı/Çakışmasız Rota Gösterimi

Çizelge 4.23. Test-12 için Heterojen On OTA'nın Düğümlere Giriş Çıkış Zamanları

OTA	$DÜGÜM^{Düğüme\_gelis\_zamani(t_g)}_{Düğümden\_cikis\_zamani(t_c)}$
OTA_1	$3^{(0.00)}_{(0.01)} \rightarrow 9^{(0.41)}_{(0.62)} \rightarrow 15^{(1.02)}_{(1.03)} \rightarrow 21^{(1.43)}_{(1.44)} \rightarrow 27^{(1.84)}_{(-)}$
OTA_2	$9^{(0.00)}_{(0.01)} \rightarrow 15^{(1.01)}_{(-)}$
OTA_3	$27^{(0.00)}_{(0.01)} \rightarrow 21^{(0.51)}_{(0.52)} \rightarrow 20^{(1.02)}_{(1.03)} \rightarrow 14^{(1.53)}_{(1.54)} \rightarrow 8^{(2.04)}_{(2.05)} \rightarrow 2^{(2.55)}_{(2.56)} \rightarrow 3^{(3.06)}_{(-)}$
OTA_4	$25^{(0.00)}_{(0.01)} \rightarrow 26^{(0.505)}_{(0.515)} \rightarrow 27^{(1.03)}_{(1.04)} \rightarrow 28^{(1.535)}_{(-)}$
OTA_5	$26^{(0.00)}_{(0.01)} \rightarrow 27^{(1.01)}_{(-)}$
OTA_6	$34^{(0.00)}_{(0.01)} \rightarrow 28^{(2,01)}_{(-)}$
OTA_7	$4^{(0.00)}_{(0.01)} \rightarrow 10^{(0.503)}_{(0.513)} \rightarrow 16^{(1.00)}_{(1.01)} \rightarrow 22^{(1.51)}_{(1.52)} \rightarrow 28^{(2.03)}_{(2.04)} \rightarrow 34^{(2.53)}_{(-)}$
OTA_8	$18^{(0.00)}_{(0.01)} \rightarrow 17^{(0.503)}_{(0.513)} \rightarrow 16^{(1.02)}_{(1.03)} \rightarrow 15^{(1.53)}_{(1.54)} \rightarrow 14^{(2.03)}_{(2.04)} \rightarrow 13^{(2.53)}_{(-)}$
OTA_9	$12^{(0.00)}_{(0.01)} \rightarrow 11^{(0.41)}_{(0.42)} \rightarrow 10^{(0.82)}_{(0.83)} \rightarrow 9^{(1.23)}_{(1.24)} \rightarrow 8^{(1.64)}_{(-)}$
OTA_{10}	$11^{(0.00)}_{(0.01)} \rightarrow 5^{(2.01)}_{(2.02)} \rightarrow 4^{(4.02)}_{(4.03)} \rightarrow 3^{(6.03)}_{(6.04)} \rightarrow 9^{(8.04)}_{(-)}$

Bu durumda önerilen algoritma OTA_7 'yi 22-28 kenarında (Şekil 4.12'e kesikli turkuaz çizgili dikdörtgen ile gösterilmiştir) OTA_6 28 numaralı düğümü terk edene kadar bekletmiştir. 28 numaralı düğüm OTA_6 'nın bitiş düğümü olduğu için OTA_7 'yi bekletmek daha az maliyetlidir. Bu sebeple algoritma OTA_7 'yi bekletmiştir. OTA_7 ve OTA_8 arasında 16 numaralı düğümde farklı doğrultuda düğüm üzerinde karşılıklı çakışma durumu meydana gelmektedir. Önerilen algoritma çakışmayı önlemek için OTA_8 'i 16-17 kenarında (Şekil 4.12'de kesikli açık yeşil çizgili dikdörtgen ile gösterilmiştir) OTA_7 16 numaralı düğümü terk edene kadar bekletmiştir. OTA_9 ve OTA_{10} arasında 10-11 kenarı üzerinde hız farkından kaynaklanan arka arkaya kenar çakışması durumu meydana gelmektedir.

Önerilen algoritma bu durumda OTA_{10} 'un rotasını değiştirmiştir. OTA_1 ve OTA_2 arasında 9-15 kenarı üzerinde hız farkından kaynaklanan arka arkaya kenar çakışması durumu meydana gelmektedir. 15 numaralı düğüm OTA_2 'nin son düğümüdür. Bu durumda önerilen algoritma OTA_1 'i 9 numaralı düğümde (Şekil 4.12'de kesikli turuncu çizgili daire ile gösterilmiştir) OTA_2 15 numaralı düğümü terk edene kadar bekletmiştir. OTA_1 ve OTA_3 arasında 15-21 kenarında kenar üzerinde karşılıklı çakışma durumu meydana gelmektedir. Önerilen algoritma bu durumda OTA_3 'ün rotasını değiştirmiştir. Şekil 4.12'de OTA_3, OTA_3 'un çakışma olmaması durumunda gidecekleri rotalar kesik çizgili ok (sırasıyla turkuaz, yeşil) ile gösterilirken önerilen algoritmanın oluşturduğu rotalar düz okla (sırasıyla turkuaz, yeşil) gösterilmiştir.

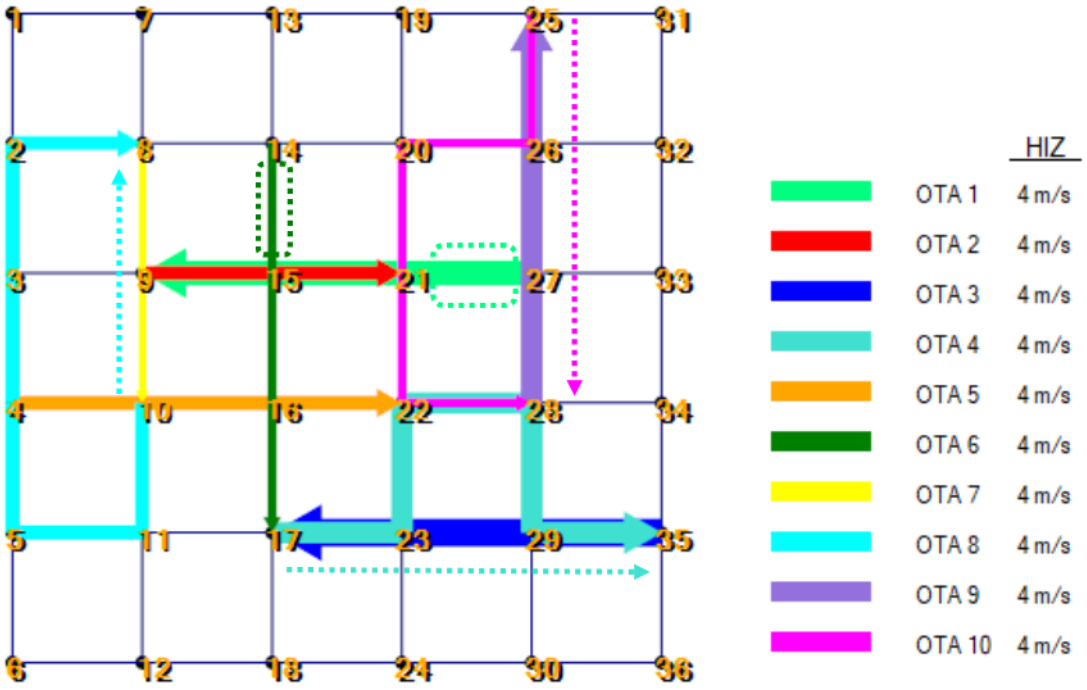
Çizelge 4.23 tüm OTA'lar için önerilen algoritmanın bulduğu rotaları ve bu rotalar üzerindeki her düğüme giriş-çıkış zamanlarını göstermektedir. Çizelge 4.24 her OTA için çakışmaların dikkate alındığı ve alınmadığı rota uzunlukları ile rota sürelerini göstermektedir. Buna göre çakışmalar dikkate alınmadan hesaplanan toplam rota uzunluğu 120m, süre 19.91s iken, önerilen algoritmanın çakışmaları çözerek bulduğu rota uzunluğu 136m, süre 25.20s'dir.

Çizelge 4.24. Test-12 için Heterojen On OTA'nın Çakışmalı/Çakışmasız Rota Uzunluğu ve Rota Tamamlanma Süreleri

OTA	Çakışmanın Dikkate Alınmadığı Rota Uzunluğu	Çakışmanın Dikkate Alındığı Rota Uzunluğu	Çakışmanın Dikkate Alınmadığı Rota Süresi	Çakışmanın Dikkate Alındığı Rota süresi
OTA_1	16m	16m	1.64s	1.84s
OTA_2	4m	4m	1.01s	1.01s
OTA_3	16m	24m	2.04s	3.06s
OTA_4	12m	12m	1.525s	1.535s
OTA_5	4m	4m	1.01s	1.01s
OTA_6	4m	4m	2.01s	2.01s
OTA_7	20m	20m	2.51s	2.53s
OTA_8	20m	20m	2.51s	2.53s
OTA_9	16m	16m	1.64s	1.64s
OTA_{10}	8m	16m	4.02s	8.04s
Maksimum	*20m*	*24m*	*4.02s*	*8.04s*

Şekil 4.13'te görülen Test-13 aynı ortamda hareket eden homojen on OTA için oluşturulmuştur. Ortamdaki tüm OTA'ların başlangıç ve bitiş düğümleri Çizelge 4.26'da verilmiştir. Bu testte üç adet çakışma tipine ait toplam yedi çakışma meydana gelmiştir. Önerilen algoritma bu çakışma tiplerinin tamamını aynı test içerisinde çözebilmiştir. OTA_7 ve OTA_8 arasında 9 numaralı düğümde aynı doğrultuda düğüm üzerinde karşılıklı çakışma durumu meydana gelmektedir. Önerilen algoritma bu durumda OTA_8 'in rotasını değiştirmiştir. Burada önerilen algoritma OTA_8 'i ilk olarak 4 numaralı düğüme yönlendirmiştir. Bu durumda OTA_5 ile OTA_8 arasında 4-10 kenarı üzerinde kenar üzerinde karşılıklı çakışma durumu meydana gelmektedir. Önerilen algoritma bu sebeple bu kez OTA_8 'i 10-16-15 yönlendirmiştir. Bu durumda ise OTA_8 ile OTA_6 arasında 15-16 kenarında kenar üzerinde karşılıklı çakışma durumu meydana gelmektedir. Sonuç olarak önerilen algoritma OTA_8 'i 10-11 rotası üzerinden bitiş düğümüne yönlendirmiştir. OTA_3 ile OTA_4 arasında 23-29 kenarı üzerinde kenar üzerinde karşılıklı çakışma durumu meydana gelmektedir. Önerilen algoritma bu durumda OTA_4 'ün rotasını değiştirmiştir. OTA_9 ile OTA_{10} arasında 26-27 kanarı üzerinde kenar üzerinde karşılıklı çakışma durumu meydana gelmektedir. Önerilen algoritma bu durumda OTA_{10} 'un rotasını değiştirmiştir. Şekil 4.13'te OTA_4, OTA_8, OTA_{10}

'un çakışma olmaması durumunda gidecekleri rotalar kesik çizgili ok (sırasıyla mavi, turkuaz, pembe) ile gösterilirken önerilen algoritmanın oluşturduğu rotalar düz okla (sırasıyla mavi, turkuaz, pembe) gösterilmiştir. OTA_2 ve OTA_6 arasında 15 numaralı düğüm üzerinde farklı doğrultuda düğüm üzerinde karşılıklı çakışma durumu meydana gelmektedir. Önerilen algoritma bu durumda OTA_6'yı 14-15 kenarı üzerinde (Şekil 4.13'te kesikli yeşil çizgili dikdörtgen ile gösterilmiştir) OTA_2 15 numaralı düğümü terk edene kadar bekletmiştir. OTA_1 ve OTA_2 arasında 15-21 kenarı üzerinde kenar üzerinde karşılıklı çakışma durumu meydana gelmiştir. Bu durumda önerilen algoritma OTA_1'i 21-27 kenarında (Şekil 4.13'te kesikli açık yeşil çizgili dikdörtgen ile gösterilmiştir) OTA_2 21 numaralı düğümü terk edene kadar bekletmiştir. 21 numaralı düğüm OTA_2'nin bitiş düğümü olduğu için OTA_1'i bekletmek daha az maliyetlidir. Bu sebeple algoritma OTA_1'i bekletmiştir.



Şekil 4.13. Homojen On OTA için Yapılan Test-13'ün Çakışmalı/Çakışmasız Rota Gösterimi

Çizelge 4.25 tüm OTA'lar için önerilen algoritmanın bulduğu rotaları ve bu rotalar üzerindeki her düğüme giriş-çıkış zamanlarını göstermektedir. Çizelge 4.26 her OTA için çakışmaların dikkate alındığı ve alınmadığı rota uzunlukları ile rota sürelerini göstermektedir. Buna göre çakışmalar dikkate alınmadan hesaplanan toplam rota uzunluğu

108m, süre 27.27s iken, önerilen algoritmanın çakışmaları çözerek bulduğu rota uzunluğu 140m, süre 36.40s'dir.

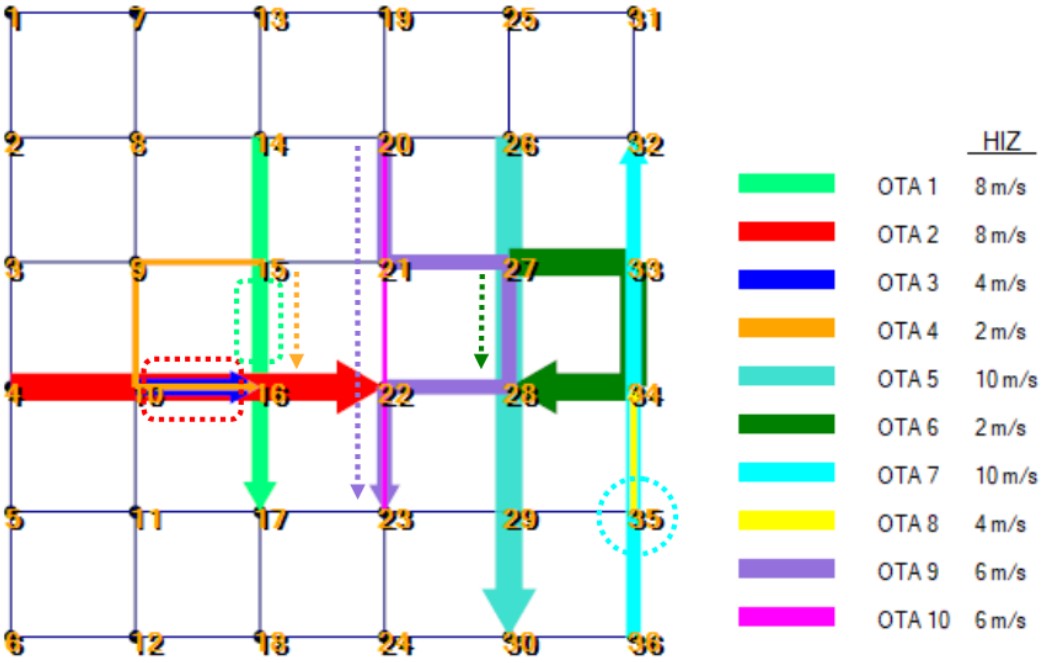
Çizelge 4.25. Test-13 İçin Homojen On OTA'nın Düğümlere Giriş Çıkış Zamanları

OTA	$DÜGÜM^{Dügüme\_gelis\_zamani(t_g)}_{Dügümden\_cikis\_zamani(t_c)}$
OTA_1	$27^{(0.00)}_{(0.01)} \rightarrow 21^{(2.04)}_{(2.05)} \rightarrow 15^{(3.05)}_{(3.06)} \rightarrow 9^{(4.06)}_{(-)}$
OTA_2	$9^{(0.00)}_{(0.01)} \rightarrow 15^{(1.01)}_{(1.02)} \rightarrow 15^{(2.02)}_{(-)}$
OTA_3	$35^{(0.00)}_{(0.01)} \rightarrow 29^{(1.01)}_{(1.02)} \rightarrow 23^{(2.02)}_{(2.03)} \rightarrow 17^{(3.03)}_{(-)}$
OTA_4	$17^{(0.00)}_{(0.01)} \rightarrow 23^{(1.01)}_{(1.02)} \rightarrow 22^{(2.02)}_{(2.03)} \rightarrow 28^{(3.03)}_{(3.04)} \rightarrow 29^{(4.04)}_{(4.05)} \rightarrow 35^{(5.05)}_{(-)}$
OTA_5	$4^{(0.00)}_{(0.01)} \rightarrow 10^{(1.01)}_{(1.02)} \rightarrow 16^{(2.02)}_{(2.03)} \rightarrow 22^{(3.03)}_{(-)}$
OTA_6	$14^{(0.00)}_{(0.01)} \rightarrow 15^{(1.03)}_{(1.04)} \rightarrow 16^{(2.04)}_{(2.05)} \rightarrow 17^{(3.05)}_{(-)}$
OTA_7	$8^{(0.00)}_{(0.01)} \rightarrow 9^{(1.01)}_{(1.02)} \rightarrow 10^{(2.02)}_{(-)}$
OTA_8	$10^{(0.00)}_{(0.01)} \rightarrow 11^{(1.01)}_{(1.02)} \rightarrow 5^{(2.02)}_{(2.03)} \rightarrow 4^{(3.03)}_{(3.04)} \rightarrow 3^{(4.04)}_{(4.05)} \rightarrow 2^{(5.05)}_{(5.06)} \rightarrow 8^{(6.06)}_{(-)}$
OTA_9	$28^{(0.00)}_{(0.01)} \rightarrow 27^{(1.01)}_{(1.02)} \rightarrow 26^{(2.02)}_{(2.03)} \rightarrow 25^{(3.03)}_{(-)}$
OTA_{10}	$25^{(0.00)}_{(0.01)} \rightarrow 26^{(1.01)}_{(1.02)} \rightarrow 20^{(2.02)}_{(2.03)} \rightarrow 21^{(3.03)}_{(3.04)} \rightarrow 22^{(4.04)}_{(4.05)} \rightarrow 28^{(5.05)}_{(-)}$

Çizelge 4.26. Test-13 için Homojen On OTA'nın Çakışmalı/Çakışmasız Rota Uzunluğu ve Rota Tamamlanma Süreleri

OTA	**Çakışmanın Dikkate Alınmadığı Rota Uzunluğu**	**Çakışmanın Dikkate Alındığı Rota Uzunluğu**	**Çakışmanın Dikkate Alınmadığı Rota Süresi**	**Çakışmanın Dikkate Alındığı Rota süresi**
OTA_1	12m	12m	3.03s	4.06s
OTA_2	8m	8m	2.02s	2.02s
OTA_3	12m	12m	3.03s	3.03s
OTA_4	12m	20m	3.03s	3.05s
OTA_5	12m	12m	3.03s	3.03s
OTA_6	12m	12m	3.03s	3.05s
OTA_7	8m	8m	2.02s	2.02s
OTA_8	8m	24m	2.02s	6.06s
OTA_9	12m	12m	3.03s	3.03s
OTA_{10}	12m	20m	3.03s	3.05s
Toplam	*12m*	*24m*	*3.03s*	*6.06s*

Şekil 4.14'de görülen Test-14 aynı ortamda hareket eden heterojen on OTA için oluşturulmuştur. Ortamdaki tüm OTA'ların başlangıç ve bitiş düğümleri Çizelge 4.28'de verilmiştir. Bu testte beş adet çakışma tipine ait toplam altı çakışma meydana gelmiştir. Önerilen algoritma bu çakışma tiplerinin tamamını aynı test içerisinde çözebilmiştir. OTA_7 ve OTA_8 arasında 34-35 kenarı üzerinde hız farkından kaynaklanan arka arkaya kenar çakışması durumu meydana gelmektedir.



Şekil 4.14. Heterojen On OTA için Yapılan Test-14'ün Çakışmalı/Çakışmasız Rota Gösterimi

Önerilen algoritma bu durumda OTA_7 'yi 35 numaralı düğüm üzerinde (Şekil 4.14'de kesikli turkuaz çizgili daire ile gösterilmiştir) OTA_8 34 numaralı düğümü terk edene kadar bekletmiştir. OTA_2 ve OTA_3 arasında 16 numaralı düğümde hız farkından kaynaklanan arka arkaya düğüm çakışması durumu meydana gelmiştir. Bu durumda önerilen algoritma OTA_2 'yi 10-16 kenarında (Şekil 4.14'de kesikli kırmızı çizgili dikdörtgen ile gösterilmiştir) OTA_3 16 numaralı düğümü terk edene kadar bekletmiştir. 16 numaralı düğüm OTA_3 'ün bitiş düğümü olduğu için OTA_2 'yi bekletmek daha az maliyetlidir. Bu sebeple algoritma OTA_2 'yi bekletmiştir. OTA_1 ile OTA_2 ve OTA_3 arasında 16 numaralı düğümde farklı doğrultuda düğüm üzerinde karşılıklı çakışma durumu meydana gelmektedir. Önerilen algoritma bu durumda

15-16 kenarı üzerinde (Şekil 4.14'de kesikli açık yeşil çizgili dikdörtgen ile gösterilmiştir) OTA_2 ve OTA_3 16 numaralı düğümü terk edene kadar bekletmiştir. OTA_1 ve OTA_4 arasında 15-16 kenarı üzerinde hız farkından kaynaklanan arka arkaya kenar çakışması durumu meydana gelmektedir. Önerilen algoritma bu durumda OTA_4 'ün rotasını değiştirmiştir. OTA_9 ve OTA_{10} arasında 21-22 kenarı üzerinde kenar üzerinde karşılıklı çakışma durumu meydana gelmektedir. Önerilen algoritma bu durumda OTA_9 'un rotasını değiştirmiştir. OTA_5 ve OTA_6 arasında 27-28 kenarı üzerinde hız farkından kaynaklanan arka arkaya kenar çakışması durumu meydana gelmiştir.

Çizelge 4.27. Test-14 İçin Heterojen On OTA'nın Düğümlere Giriş Çıkış Zamanları

OTA	$DÜGÜM^{\substack{Dügüme\_gelis\_zamani(t_g)\\ Dügümden\_cikis\_zamani(t_c)}}$
OTA_1	$14^{(0.00)}_{(0.01)} \rightarrow 15^{(0.51)}_{(0.52)} \rightarrow 16^{(1.05)}_{(1.06)} \rightarrow 17^{(1.56)}_{(-)}$
OTA_2	$4^{(0.00)}_{(0.01)} \rightarrow 10^{(0.51)}_{(0.52)} \rightarrow 16^{(1.03)}_{(1.04)} \rightarrow 22^{(1.54)}_{(-)}$
OTA_3	$10^{(0.00)}_{(0.01)} \rightarrow 16^{(1.01)}_{(-)}$
OTA_4	$15^{(0.00)}_{(0.01)} \rightarrow 9^{(2.01)}_{(2.02)} \rightarrow 10^{(4.02)}_{(4.03)} \rightarrow 16^{(6.03)}_{(-)}$
OTA_5	$26^{(0.00)}_{(0.01)} \rightarrow 27^{(0.41)}_{(0.42)} \rightarrow 28^{(0.82)}_{(0.83)} \rightarrow 29^{(1.23)}_{(1.24)} \rightarrow 30^{(1.64)}_{(-)}$
OTA_6	$27^{(0.00)}_{(0.01)} \rightarrow 33^{(2.01)}_{(2.02)} \rightarrow 34^{(4.02)}_{(4.03)} \rightarrow 28^{(6.03)}_{(-)}$
OTA_7	$36^{(0.00)}_{(0.01)} \rightarrow 35^{(0.41)}_{(0.62)} \rightarrow 34^{(1.02)}_{(1.03)} \rightarrow 33^{(1.43)}_{(1.44)} \rightarrow 32^{(1.84)}_{(-)}$
OTA_8	$35^{(0.00)}_{(0.01)} \rightarrow 34^{(1.01)}_{(-)}$
OTA_9	$20^{(0.00)}_{(0.01)} \rightarrow 21^{(0.676)}_{(0.686)} \rightarrow 27^{(1.353)}_{(1.363)} \rightarrow 28^{(2.03)}_{(2.04)} \rightarrow 22^{(2.706)}_{(2.716)} \rightarrow 23^{(3.383)}_{(-)}$
OTA_{10}	$23^{(0.00)}_{(0.01)} \rightarrow 22^{(0.676)}_{(0.686)} \rightarrow 21^{(1.353)}_{(1.363)} \rightarrow 20^{(2.03)}_{(-)}$

Önerilen algoritma bu durumda çakışmayı önlemek için OTA_6 'nın rotasını değiştirmiştir. Şekil 4.15'de OTA_4, OTA_6, OTA_9 'un çakışma olmaması durumunda gidecekleri rotalar kesik çizgili ok (sırasıyla turuncu, yeşil, mor) ile gösterilirken önerilen algoritmanın oluşturduğu rotalar düz okla (sırasıyla turuncu, yeşil, mor) gösterilmiştir.

Çizelge 4.27 tüm OTA'lar için önerilen algoritmanın bulduğu rotaları ve bu rotalar üzerindeki her düğüme giriş-çıkış zamanlarını göstermektedir. Çizelge 4.28 her OTA için çakışmaların dikkate alındığı ve alınmadığı rota uzunlukları ile rota sürelerini

göstermektedir. Buna göre çakışmalar dikkate alınmadan hesaplanan toplam rota uzunluğu 96m, süre 16.44s iken, önerilen algoritmanın çakışmaları çözerek bulduğu rota uzunluğu 120m, süre 26.073s'dir.

Çizelge 4.28. Test-14 için Heterojen On OTA'nın Çakışmalı/Çakışmasız Rota Uzunluğu ve Rota Tamamlanma Süreleri

OTA	Çakışmanın Dikkate Alınmadığı Rota Uzunluğu	Çakışmanın Dikkate Alındığı Rota Uzunluğu	Çakışmanın Dikkate Alınmadığı Rota Süresi	Çakışmanın Dikkate Alındığı Rota süresi
OTA_1	12m	12m	1.53s	1.56s
OTA_2	12m	12m	1.53s	1.54s
OTA_3	4m	4m	1.01s	1.01s
OTA_4	4m	12m	2.01s	6.03s
OTA_5	16m	16m	1.64s	1.64s
OTA_6	4m	12m	2.01s	6.03s
OTA_7	16m	16m	1.64s	1.84s
OTA_8	4m	4m	1.01s	1.01s
OTA_9	12m	20m	2.03s	3.838s
OTA_{10}	12m	12m	2.03s	2.03s
Maksimum	*16m*	*20m*	*2.03s*	*6.03s*

On OTA için yapılan testlerde çakışma durumu dikkate alınarak hesaplanan rota uzunluğu ve süresi, çakışma durumu dikkate alınmadan hesaplanan rota uzunluğu ve süresinden daha fazladır. Ancak çakışma durumu düşünülmeden OTA'lar harekete başladığı takdirde meydana gelen anlık çakışma durumlarında OTA'lar çakışmanın meydana geldiği OTA'yı rotası tamamlanana kadar beklemek zorunda kalacaktı. Karşılıklı kenar çakışması veya aynı doğrultuda karşılıklı düğüm çakışması durumları meydana geldiğinde her iki OTA'da karşılıklı olarak birbirini bekleyecek ve kilitlenme meydana gelecekti. Özellikle ortamdaki OTA sayısı arttıkça çakışma ihtimali de artacağı için OTA'lar arasında ardışıl çakışmalar meydana gelebilmektedir. Çakışma durumu başlangıçta düşünülmediği zaman çakışmadan kaynaklı gereksiz beklemeler düğümleri fazladan işgal edebilir ve bu durumda daha fazla çakışmaya sebep olabilir. Ortamdaki gereksiz beklemeleri azaltmak ve kilitlenmeleri önlemek açısından çakışmaların başlangıçta hesaba katılarak uygun rotaların oluşturması son derece önemlidir.

5. SONUÇ VE ÖNERİLER

Bu çalışmada akıllı fabrikalarda çoklu OTA'ların çakışmasız rotalanması problemi çalışılmıştır. Problemi çözmek için A* algoritması temeline dayanan bir algoritma önerilmiştir. Tek bir başlangıç ve bitiş noktası arasında en kısa rotayı (yolu) bulmayı amaçlayan A* algoritması, önerilen algoritma da birden fazla başlangıç ve bitiş düğümü için çalışacak şekilde geliştirilmiştir. Önerilen algoritma rotaları oluştururken meydana gelebilecek çakışmaları anlayıp, mümkün olan en optimum şekilde çözüm üretebilmektedir.

Önerilen yöntem farklı sayıda OTA için verilen ortamda test edilmiştir. Yapılan testlerde araç sayısı arttıkça aynı OTA için ardışıl olarak birden fazla çakışma durumu oluşabilmektedir. Önerilen algoritma bu durumda aynı OTA için meydana gelen tüm ardışıl çakışmaları ayrı ayrı tespit ederek bu çakışmaları sırasıyla çözerek uygun çakışmasız rotayı hesaplayabilmektedir. Heterojen veya homojen on OTA için yapılan testlerde çalışmada bahsedilen çakışma tiplerine ait çok sayıda çakışma meydana gelmiştir. Test sonuçlarına bakıldığında önerilen algoritmanın meydana gelen tüm çakışma tiplerini tespit ederek çakışma tipine göre optimum maliyetli çözüm stratejisini seçerek çakışmasız uygun rotaları hesaplayabildiği görülmektedir. Bu durum önerilen algoritmanın başarılı bir şekilde çalıştığını göstermektedir.

Tez kapsamındaki çalışmada OTA'ların hızı birbirinden farklı olabilirken aynı OTA'nın hızı hareket süresi boyunca sabit kabul edilmiştir. Ortamda hareket ederken yollardaki dönme noktalarını da dikkate alarak yol boyunca değişken hızlara sahip OTA'lar için çakışmasız rotalar oluşturma önemli bir çalışma konusu olabilir. Bir başka çalışma konusu olarak fabrika içi çizelgeleme ve çakışmasız rota planlama problemi beraber olarak ele alınabilir. Çizelgeleme ve rotalama probelemini çözmek için optimum sayıda kullanılması gereken OTA sayısı hesaplanarak, belirlenen sayıdaki OTA için çakışmasız rotalar hesaplanabilir.

KAYNAKLAR DİZİNİ

Adamo, T., Bektas, T., Ghiani, G., Guerriero, E., & Manni, E. (2018). Path and speed optimization for conflict-free pickup and delivery under time windows. *Transportation Science*, *52*(4), 739–755. https://doi.org/10.1287/trsc.2017.0816

Alpalsan, M. (2015). *Araç Rotalama Problemleri İçin Matematiksel Modeller Ve Çözüm Yöntemleri*.

Aslan, Ö., & Yazici, A. (2020). Otonom Taşıyıcı Araçlar için Çakışmasız Rota Planlama. *28. IEEE SİNYAL İŞLEME ve İLETİŞİM UYGULAMALARI KURULTAYI, Basım Aşamasında*, 1–4.

Bae, J., & Chung, W. (2018). A Heuristic for Path Planning of Multiple Heterogeneous Automated Guided Vehicles. *International Journal of Precision Engineering and Manufacturing*, *19*(12), 1765–1771. https://doi.org/10.1007/s12541-018-0205-x

Chen, T., Sun, Y., Dai, W., Tao, W., & Liu, S. (2013). On the shortest and conflict-free path planning of multi-agv systembased on dijkstra algorithm and the dynamic time-window method. *Advanced Materials Research*, *645*, 267–271. https://doi.org/10.4028/www.scientific.net/AMR.645.267

Cui, Y., Ma, D.-P., Fang, Y., & Lei, Z. (2019). Conflict-Free Path Planning of Agv Based on Improved a-Star Algorithm. *Recent Developments on Information and Communication Technology (ICT) Engineering*, 31–34. https://doi.org/10.35745/icice2018v2.008

Desaulniers, G., Langevin, A., Riopel, D., & Villeneuve, B. (2003). Dispatching and conflict-free routing of automated guided vehicles: An exact approach. *International Journal of Flexible Manufacturing Systems*, *15*(4), 309–331. https://doi.org/10.1023/B:FLEX.0000036033.12568.5e

Ercan, C., & Gencer, C. (2013). İnsansız Hava Sistemleri Rota Planlaması Dinamik Çözüm Metotları Ve Literatür Araştırması. *Selcuk Univ. J. Eng. Sci. Tech.*, *1*(2), 51–72. https://doi.org/10.1017/CBO9781107415324.004

Giordani, S., Lujak, M., & Martinelli, F. (2013). A distributed multi-agent production planning and scheduling framework for mobile robots. *Computers and Industrial Engineering*, *64*(1), 19–30. https://doi.org/10.1016/j.cie.2012.09.004

Goyal, A., Mogha, P., Luthra, R., & Sangwan, N. (2014). Path Finding: a* or Dijkstra'S? *International Journal in IT and Engineering*, *02*(01).

Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). Formal Basis for the Heuristic Determination eijj ,. *Systems Science and Cybernetics*, *4*(2), 100–107.

Herrero, D., & Mart, H. (2010). Modeling Distributed Transportation Systems Composed of Flexible Automated Guided Vehicles in Flexible Manufacturing Systems. *IEEE Transactions on Industrial Informatics*, *6*(2), 166–180.

KAYNAKLAR DİZİNİ (devam)

Hussein, A., Mostafa, H., Badrel-Din, M., Sultan, O., & Khamis, A. (2012). Metaheuristic optimization approach to mobile robot path planning. *International Conference on Engineering and Technology, ICET 2012 - Conference Booklet*. https://doi.org/10.1109/ICEngTechnol.2012.6396150

Jia, F., Ren, C., Chen, Y., & Xu, Z. (2017). A system control strategy of a conflict-free multi-AGV routing based on improved A∗ algorithm. *2017 24th International Conference on Mechatronics and Machine Vision in Practice, M2VIP 2017*, 1–6. https://doi.org/10.1109/M2VIP.2017.8211447

Kim, C. W., & Tanchoco, J. M. A. (1991). Conflict-free shortest-time bidirectional AGV routeing. *International Journal of Production Research*, Vol. 29, pp. 2377–2391. https://doi.org/10.1080/00207549108948090

Kim, S., Jin, H., Seo, M., & Har, D. (2019). Optimal Path Planning of Automated Guided Vehicle using Dijkstra Algorithm under Dynamic Conditions. *2019 7th International Conference on Robot Intelligence Technology and Applications, RiTA 2019*, 231–236. https://doi.org/10.1109/RITAPP.2019.8932804

Krishnamurthy, N. N., Batta, R., & Karwan, M. H. (1993). Developing Conflict-Free Routes for Automated Guided Vehicles. *Operations Research*, *41*(6), 1077–1090.

Langevin, A., Lauzon, D., & Riopel, D. (1996). Dispatching, routing, and scheduling of two automated guided vehicles in a flexible manufacturing system. *International Journal of Flexible Manufacturing Systems*, *8*(3), 247–262. https://doi.org/10.1007/BF00403127

Li, J. J., Xu, B. W., Postolache, O., Yang, Y. S., & Wu, H. F. (2018). Impact Analysis of Travel Time Uncertainty on AGV Catch-Up Conflict and the Associated Dynamic Adjustment. *Mathematical Problems in Engineering*, *2018*. https://doi.org/10.1155/2018/4037695

Lin, M., Yuan, K., Shi, C. J., & Wang, Y. (2017). Path Planning of Mobile Robot Based on Improved A * Algorithm. *29th Chinese Control And Decision Conference (CCDC)*, 3570–3576.

Liu, C., Tan, J., Zhao, H., Li, Y., & Bai, X. (2017). Path planning and intelligent scheduling of multi-AGV systems in workshop. *36th Chinese Control Conference, CCC*, 2735–2739. https://doi.org/10.23919/ChiCC.2017.8027778

Lyu, X., Song, Y., He, C., Lei, Q., & Guo, W. (2019). Approach to Integrated Scheduling Problems Considering Optimal Number of Automated Guided Vehicles and Conflict-Free Routing in Flexible Manufacturing Systems. *IEEE Access*, *7*, 74909–74924. https://doi.org/10.1109/ACCESS.2019.2919109

KAYNAKLAR DİZİNİ (devam)

Małopolski, W. (2018). A sustainable and conflict-free operation of AGVs in a square topology. *Computers and Industrial Engineering*, *126*(October), 472–481. https://doi.org/10.1016/j.cie.2018.10.002

Maza, S., & Castagna, P. (2001). Conflict-free AGV routing in bi-directional network. *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, *2*(c), 761–764. https://doi.org/10.1109/etfa.2001.997777

Maza, S., & Castagna, P. (2005). Sequence based hierarchical conflict-free routing strategy of bi-directional automated guided vehicles. *IFAC Proceedings Volumes (IFAC-PapersOnline)*, *16*(c), 168–173.

Miyamoto, T., & Inoue, K. (2016). Local and random searches for dispatch and conflict-free routing problem of capacitated AGV systems. *Computers and Industrial Engineering*, *91*, 1–9. https://doi.org/10.1016/j.cie.2015.10.017

Möhring, R. H., Köhler, E., Gawrilow, E., & Stenzel, B. (2005). Conflict-free Real-time AGV Routing. *Operations Research Proceedings 2004*, (January), 18–24. https://doi.org/10.1007/3-540-27679-3

Murakami, K. (2020). Time-space network model and MILP formulation of the conflict-free routing problem of a capacitated AGV system. *Computers and Industrial Engineering*, *141*(November 2019), 106270. https://doi.org/10.1016/j.cie.2020.106270

Nishi, T., Hiranaka, Y., & Grossmann, I. E. (2011). A bilevel decomposition algorithm for simultaneous production scheduling and conflict-free routing for automated guided vehicles. *Computers & Operations Research*, *38*, 876–888. https://doi.org/10.1016/j.cor.2010.08.012

Nishi, T., & Tanaka, Y. (2012). Petri net decomposition approach for dispatching and conflict-free routing of bidirectional automated guided vehicle systems. *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans*, *42*(5), 1230–1243. https://doi.org/10.1109/TSMCA.2012.2183353

Oboth, C., Batta, R., & Karwan, M. (1999). Dynamic conflict-free routing of automated guided vehicles. *International Journal of Production Research*, *37*(9), 2003–2030. https://doi.org/10.1080/002075499190888

Qing, G., Zheng, Z., & Yue, X. (2017). Path-planning of automated guided vehicle based on improved Dijkstra algorithm. *Proceedings of the 29th Chinese Control and Decision Conference, CCDC 2017*, 7138–7143. https://doi.org/10.1109/CCDC.2017.7978471

KAYNAKLAR DİZİNİ (devam)

Qiu, L., Wang, J., Chen, W., & Wang, H. (2015). Heterogeneous AGV routing problem considering energy consumption. *2015 IEEE International Conference on Robotics and Biomimetics, IEEE-ROBIO 2015*, 1894–1899. https://doi.org/10.1109/ROBIO.2015.7419049

Riazi, S., Diding, T., Falkman, P., Bengtsson, K., & Lennartson, B. (2019). Scheduling and routing of agvs for large-scale flexible manufacturing systems. *2019 IEEE 15th International Conference on Automation Science and Engineering*, *2019-Augus*, 891–896. https://doi.org/10.1109/COASE.2019.8842849

Saidi-Mehrabad, M., Dehnavi-Arani, S., Evazabadian, F., & Mahmoodian, V. (2015). An Ant Colony Algorithm (ACA) for solving the new integrated model of job shop scheduling and conflict-free routing of AGVs. *Computers and Industrial Engineering*, *86*, 2–13. https://doi.org/10.1016/j.cie.2015.01.003

Smolic-rocak, N., Bogdan, S., Kovacic, Z., & Petrovic, T. (2010). Time Windows Based Dynamic Routing in Multi-AGV Systems. *IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING*, *7*(1), 151–155.

Sun, X., Zhao, Y., Shen, S., Wang, K., Zheng, X., & Shi, Y. (2018). Scheduling Multiple AGVs with Dynamic Time-windows for Smart Indoor Parking Lot. *Proceedings of the 2018 IEEE 22nd International Conference on Computer Supported Cooperative Work in Design, CSCWD 2018*, 57–62. https://doi.org/10.1109/CSCWD.2018.8465303

Tai, R., Wang, J., Tian, W., Chen, W., Wang, H., & Zhou, Y. (2019). A time-efficient approach to solve conflicts and deadlocks for scheduling AGVs in warehousing applications. *2018 IEEE International Conference on Real-Time Computing and Robotics, RCAR 2018*, 166–171. https://doi.org/10.1109/RCAR.2018.8621773

ter Mors, A. W. (2011). Conflict-free route planning in dynamic environments. *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, (1), 2166–2171. https://doi.org/10.1109/iros.2011.6094461

Thanos, E., Wauters, T., & Vanden Berghe, G. (2019). Dispatch and conflict-free routing of capacitated vehicles with storage stack allocation. *Journal of the Operational Research Society*, 1–14. https://doi.org/10.1080/01605682.2019.1595191

Thomas, S., Deodhare, D., & Narasimha Murty, M. (2019). Extended conflict-based search with awareness. *Integrated Intelligent Computing, Communication and Security*, *771*, 459–467. https://doi.org/10.1007/978-981-10-8797-4_47

KAYNAKLAR DİZİNİ (devam)

Tsang, K. F. E., Ni, Y., Wong, C. F. R., & Shi, L. (2018). A Novel Warehouse Multi-Robot Automation System with Semi-Complete and Computationally Efficient Path Planning and Adaptive Genetic Task Allocation Algorithms. *2018 15th International Conference on Control, Automation, Robotics and Vision, ICARCV 2018*, 1671–1676. https://doi.org/10.1109/ICARCV.2018.8581092

Umar, U. A., Ariffin, M. K. A., Ismail, N., & Tang, S. H. (2013). Conflict-free Automated Guided Vehicles routing using multi-objective genetic algorithm. *Research Journal of Applied Sciences, Engineering and Technology*, *6*(14), 2681–2684. https://doi.org/10.19026/rjaset.6.3758

Wang, C., Wang, L., Qin, J., Wu, Z., Duan, L., Li, Z., … Wang, Q. (2015). Path planning of automated guided vehicles based on improved A-Star algorithm. *2015 IEEE International Conference on Information and Automation, ICIA 2015*, (August), 2071–2076. https://doi.org/10.1109/ICInfA.2015.7279630

Xidias, E., Zacharia, P., & Nearchou, A. (2016). Path Planning and scheduling for a fleet of autonomous vehicles. *Robotica*, *34*(10), 2257–2273. https://doi.org/10.1017/S0263574714002872

Yan, X., Zhang, C., & Qi, M. (2017). Multi-AGVs Collision-Avoidance and Deadlock-Control for Item-To-Human Automated Warehouse. *2017 International Conference on Industrial Engineering, Management Science and Application, ICIMSA 2017*, 1–5. https://doi.org/10.1109/ICIMSA.2017.7985596

Yuan, R., Dong, T., & Li, J. (2016). Research on the Collision-Free Path Planning of Multi-AGVs System Based on Improved A* Algorithm. *American Journal of Operations Research*, *06*(06), 442–449. https://doi.org/10.4236/ajor.2016.66041

Zeng, W., & Church, R. L. (2009). Finding shortest paths on real road networks: The case for A. *International Journal of Geographical Information Science*, *23*(4), 531–543. https://doi.org/10.1080/13658810801949850

Zhang, H., Luo, H., Wang, Z., Liu, Y., & Liu, Y. (2019). Multi-Robot Cooperative Task Allocation with Definite Path-Conflict-Free Handling. *IEEE Access*, *7*, 138495–138511. https://doi.org/10.1109/ACCESS.2019.2942966

Zhang, W., Peng, Y., Wei, W., & Kou, L. (2018). Real-Time Conflict-Free Task Assignment and Path Planning of Multi-AGV System in Intelligent Warehousing. *2018 37th Chinese Control Conference (CCC)*, 5311–5316.

Zhang, Y., Li, L. L., Lin, H. C., Ma, Z., & Zhao, J. (2019). Development of path planning approach using improved a-star algorithm in AGV system. *Journal of Internet Technology*, *20*(3), 915–924. https://doi.org/10.3966/160792642019052003023

KAYNAKLAR DİZİNİ (devam)

Zhang, Z., Guo, Q., Chen, J., & Yuan, P. (2018). Collision-Free Route Planning for Multiple AGVs in an Automated Warehouse Based on Collision Classification. *IEEE Access*, *6*, 26022–26035. https://doi.org/10.1109/ACCESS.2018.2819199

Zhang, Z., Guo, Q., & Yuan, P. (2017). Conflict-free route planning of automated guided vehicles based on conflict classification. *2017 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2017*, *2017-Janua*, 1459–1464. https://doi.org/10.1109/SMC.2017.8122819