Çok Gezgin Robotlu Tam Kapsama Yol Planlaması İçin Sarmal Kapsar Ağaç Tabanlı Bir Yaklaşım

Çiğdem Özbek

YÜKSEK LİSANS TEZİ

Endüstri Mühendisliği Anabilim Dalı

Haziran– 2010

Spiral Spanning Tree Based Approach For Multiple Mobile Robot Full Coverage Path Planning

Çiğdem Özbek

MASTER OF SCIENCE THESIS

Department of Industrial Engineering

June– 2010

Çok Gezgin Robotlu Tam Kapsama Yol Planlaması İçin Sarmal Kapsar Ağaç Tabanlı Bir Yaklaşım

Çiğdem Özbek

Eskişehir Osmangazi Üniversitesi
Fen Bilimleri Enstitüsü
Lisansüstü Yönetmeliği Uyarınca
Endüstri Mühendisliği Anabilim Dalı
Yöneylem Araştırması Bilim Dalında
YÜKSEK LİSANS TEZİ
Olarak Hazırlanmıştır

Danışman: Doç. Dr. Muzaffer Kapanoğlu

Haziran– 2010

ONAY

Endüstri Mühendisliği Anabilim Dalı Yüksek Lisans öğrencisi Çiğdem Özbek'in YÜKSEK LİSANS tezi olarak hazırladığı "Çok Gezgin Robotlu Tam Kapsama Yol Planlaması İçin Sarmal Kapsar Ağaç Tabanlı Bir Yaklaşım" başlıklı bu çalışma, jürimizce lisansüstü yönetmeliğin ilgili maddeleri uyarınca değerlendirilerek kabul edilmiştir.

Danışman **:** Doç. Dr. Muzaffer KAPANOĞLU

İkinci Danışman : -

Yüksek Lisans Tez Savunma Jürisi:

Üye : Doç. Dr. Osman PARLAKTUNA

Üye : Doç.Dr. Müjgan SAĞIR

Üye : Doç.Dr. Muzaffer KAPANOĞLU

Üye : Yrd.Doç.Dr. Aykut ARAPOĞLU

Üye : Yrd.Doç.Dr. Şerafettin ALPAY

Fen Bilimleri Enstitüsü Yönetim Kurulu'nun . . . . . tarih ve . . . . . sayılı kararıyla onaylanmıştır.

Prof. Dr. Nimetullah BURNAK

Enstitü Müdürü

ÖZET

Bu çalışmada, çoklu robotların tam kapsama gerçekleştirmeleri için gezinti yollarının planlanması problemi ele alınmıştır. Öncelikle, çoklu robot kapsama alanında literatürde yer alan yol planlama algoritmaları incelenmiş ve araştırma sonucu elde edilen bilgilere yer verilmiştir. Araştırma kapsamında algılayıcı tabanlı gezgin çok robotlu statik çalışma alanının tam kapsanması için yeni ve diğer yöntemlere göre avantajları söz konusu olan ÇS-KAK; Çoklu Sarmal - Kapsar Ağaç Kapsama (MS-STC, Multiple Spiral - Spanning Tree Coverage) algoritması geliştirilmiştir. İlgili algoritma Visual Studio 2008 ortamında C# programlama dili kullanılarak yazılmıştır. Geliştirilen algoritmanın robotlara eşit iş yükü paylaştırması, robotların başlangıç noktasında serbestlik sağlaması ve kapsama alanlarındaki değişimlere uyum sağlayabilmesi uygulama değerine sahip üstünlükleridir. Algoritma, bu alanda en çok bilinen ve/veya en yeni yol planlama yöntemleri ile karşılaştırılmıştır. Önerilen algoritmanın farklı problemlerdeki performansı test edilerek değerlendirilmiş, gelecekte yapılacak çalışmalara ışık tutması sağlanmıştır.

Anahtar Kelimeler: Çok Gezgin Robot, Kapsar Ağaç, Tam Kapsama Algoritması

SUMMARY

In this study, the path planning problem of multiple robot has been studied for full coverage. Firstly, path planning algorithms which are in the literature have been experimented and the information that is obtained from the result of the literature survey is presented. At the survey new MS-STC (Multiple Robot Spiral – Spanning Tree Coverage) which has advantages as per the other methods has been developed for full coverage of the static work area with sensor based mobile multiple robot. Algorithm has been coded by using the C# at the Visual Studio 2008. MS-STC advantages are to share equal work to robots, to exist freedom for robot initial locations, to increase feasibility rate for dinamic work area. The algorithm has been compared with the other most known and/or the newest path planning methods. The performance of proposed algorithm has been evaluated on different test problems, future studies are provided to shed light.

Keywords: Multiple Mobile Robot, Spanning Tree, Full Coverage Algorithm

TEŞEKKÜR

Tez konumun seçimi, yönlendirmesi ve şekillendirilmesindeki katkılarından dolayı hocam **Sayın Doç. Dr. Muzaffer Kapanoğlu**'na, bu çalışmayı gerçekleştirebilmem için yardımlarını esirgemeyen OSMANGAZİ ÜNİVERSİTESİ ENFORMATİK BÖLÜMÜ çalışanlarına, sabrı ve bana duyduğu inancı ile cesaret kaynağım olan babam **Ömer Topçu**'ya, başarımı onunda başarısıdır addettiğim moral ve desteği ile her zaman yanımda olan eşim **Erdal Özbek**'e teşekkürlerimi bir borç bilirim.

İÇİNDEKİLER

Sayfa

ÖZET ..... **v**

SUMMARY ..... **vi**

TEŞEKKÜR ..... **vii**

ŞEKİLLER DİZİNİ ..... **xii**

ÇİZELGELER DİZİNİ ..... **xiii**

1. GİRİŞ **1**

2. ROBOT TAM KAPSAMASI ÜZERİNE YAPILMIŞ ÖNCEKİ ÇALIŞMALAR **4**

2.1. Literatürdeki Gezgin Tek Robot Kapsama Algoritmaları 4

2.1.1. Kapsar ağaç tabanlı kapsama algoritmaları 4

2.1.2. Git-Gel Hücresel Ayrıştırma tabanlı kapsama algoritmaları 5

2.1.3. Üçgensel Hücre Ayrıştırma tabanlı kapsama algoritmaları 6

2.1.4. Sezgisel kapsama algoritmaları 6

2.2. Literatürdeki Gezgin Çok Robotlu Kapsama Algoritmaları 7

2.2.1. Kapsar ağaç tabanlı kapsama algoritmaları 8

2.2.2. Git-Gel Hücresel Ayrıştırma tabanlı kapsama algoritması 8

2.2.3. Sezgisel kapsama algoritmaları 8

2.2.4. Voronoi Diagramı tabanlı kapsama algoritması 9

3. ÇOK ROBOTLU TAM KAPSAMA YÖNTEMLERİ **10**

3.1. Gezgin Robotlar ile Sürekli Alan Kapsamada Kapsar Ağaç 10

İÇİNDEKİLER (devam)

Sayfa

3.1.1. Statik ortamda kapsar ağaç kapsama algoritması 10
3.1.2. Dinamik ortamda kapsar ağaç kapsama algoritması 11
3.1.3. Karıncaması kapsar ağaç kapsama algoritması 11
3.2. Geri Dönüşlü Sarmal Tam Kapsama Algoritması 12
3.3. Çoklu Robotlar İçin Kapsamada Fazlalık, Etkinlik, Sağlamlık 14
3.3.1. Geri dönüşsüz çoklu kapsar ağaç kapsama algoritması 14
3.3.2. Geri dönüşlü çoklu kapsar ağaç kapsama algoritması 15
3.4. Etkili Çoklu Robot Kapsaması İçin Kapsar Ağaç Yapılandırması 15
3.5. Çoklu Robot Kapsar Yol Planlamasına Genetik Algoritma Yaklaşımı 17
3.6. Çoklu Robot Kapsar Yol Planlamasında Kapsama Süresini En Küçükleyen Genetik Algoritma Yaklaşımı 19

4. ÇOK ROBOTLU TAM KAPSAMA PROBLEMİ İÇİN GELİŞTİRİLEN YAKLAŞIM 23

4.1. Çoklu Sarmal - Kapsar Ağaç Kapsama Algoritması Tanımı 23
4.2. Çoklu Sarmal-Kapsar Ağaç Kapsama Algoritma Adımları ve Akış Diyagramı 26
4.3. Çoklu Sarmal - Kapsar Ağaç Kapsama Algoritması C# Programı 29
4.4. Önerilen Yaklaşımın Testi 31
4.4.1. Büyük alanda Çoklu Sarmal - Kapsar Ağaç Kapsama algoritması 31
4.4.2. Dört farklı çalışma alanında Çoklu Sarmal - Kapsar Ağaç Kapsama algoritması 34

5. ÖNERİLEN YAKLAŞIMIN DİĞER YÖNTEMLERLE KARŞILAŞTIRILMASI 36

İÇİNDEKİLER (devam)

Sayfa

5.1. Etkili Çoklu Robot Kapsaması için Kapsar Ağaç Yapılandırması ile Çoklu Sarmal - Kapsar Ağaç Kapsama Algoritmasının Karşılaştırılması 36

5.2. Çoklu Kapsar Ağaç Kapsama Algoritması ile Çoklu Sarmal - Kapsar Ağaç Kapsama Algoritmasının Karşılaştırılması 37

5.3. Çoklu Robot Kapsamasında Genetik Algoritma ile Çoklu Sarmal - Kapsar Ağaç Kapsama Algoritmasının Karşılaştırılması 40

6. SONUÇ VE ÖNERİLER 45

KAYNAKLAR 46

EKLER .....

ŞEKİLLER DİZİNİ

Şekil **Sayfa**

3.1 Basit GSA algoritması....13
3.2 Üç robot için kapsar ağaç....15
3.3 Farklı ağaçların kapsama süresine etkisi....16
3.4 Komşu-disk önceliklendirme örüntüsü....17
3.5 OGA için kromozom yapısı....18
3.6 DGA için kromozom yapısı....19
3.7 (a) Örnek kromozom (b) Üç robot ve izin verilen örüntü sayısı iki iken elde edilen çözüm....20
4.1 Algoritma 1, R robot için sarmal kapsar ağaç yol oluşturma algoritması akış diyagramı....27
4.2 Algoritma 2, çoklu robotlar için enküçük kapsar ağaç tam kapsama algoritması akış diyagramı....28
4.3 ÇS-KAK algoritması C# Programı....29
4.4 İki robot için ÇS-KAK algoritması C# Programı....30
4.5 Dört farklı çalışma alanının birleşiminden oluşan büyük alan....31
4.6 Üç robot için büyük alanda ÇS-KAK algoritması....32
5.1 Olası yollar içinde [N/k] uzaklığının sağlanamadığı kapsar ağaç yolu....37
5.2 (a) Sekiz robot için ÇS-KAK algoritmasında kapsar ağaç yolu (b) Tek robot için kapsar ağaç yolu (c) Sekiz robotlu ÇKAK'da eşit hücre paylaşımı....38
5.3 (a) Sekiz robot için kapsar ağaç yolu (b) ÇKAK ile sekiz robot için kapsar ağaç yolu....39
5.4 Robotların başlangıç pozisyonu depo iken ÇS-KAK algoritması ve genetik algoritma için kapsama yolu....41
5.5 Robotlar rastgele belirlenmiş başlangıç noktasında iken ÇS-KAK algoritması ve genetik algoritma için elde edilen çözüm....43

ÇİZELGELER DİZİNİ

Çizelge **Sayfa**

5.1 Dört farklı çalışma alanında, farklı başlangıç noktalarında ÇS-KAK algortiması ve genetik algoritma ile elde edilen çözümlerde tekrar kapsanan hücre sayıları.....43

1. GİRİŞ

Modern üretim sistemlerinin tamamına yakını sabit veya gezgin robotlardan yararlanmaktadır. Özellikle bilgisayar-bütünleşik esnek üretim sistemleri, gerek robot kolları gerekse otomatik yönlendirmeli veya kendi-kendini yönlendiren araçlar kullanmaktadır. Günümüzde robotlar, insan gücünün yerini almaktan çok insanlara uygun olmayan işlerde tamamlayıcı, gelişmiş bir teknolojik araç olarak görülmektedir. İnsanların ve robotların birlikte kullanıldığı karma (hybrid) otomasyon sistemleri, insansız ve tam otomatik sistemlere göre bir çok üstünlüğe sahiptir. Özellikle ileri teknoloji ürünlerinin üretimi, nano-teknolojik ve farmasötik ürünlerin yanı sıra gıda üretiminde özellikle gezgin robotların önemli roller üstlendikleri gerçektir. Robotların insan gücüne tercih edilmesine sebep olabilecek özellikleri, işkoluna göre değişiklik göstermekle beraber, dar alanlara girebilecek kadar küçük yapılabilmeleri, çok soğuk, çok sıcak ya da zehirli kimyasal ve radyoaktivite madde yüklü ortamlarda çalışabilmeleri, programlanmış görevlerinin dışına çıkmamaları ve biyolojik kirliliğe sebep olmamalarıdır (Kapanoğlu vd., 2010).

Robot otonom ve önceden programlanmış görevleri yerine getirebilen elektro-mekanik bir cihazdır gibi birçok tanımı olmakla beraber en genel ifade ile robot; programlandığı zaman çeşitli canlı karakterleri gösterebilen makinelerdir. Robotlar doğrudan bir operatörün kontrolünde çalışabildikleri gibi bağımsız olarak bir bilgisayar programının kontrolünde de çalışabilir. Yer değiştirme yeteneğine sahip olan robotlar gezgin, yer değiştiremeyen robotlar ise sabit olarak tanımlanmaktadır. Günümüzde artık robotların daha geniş toleranslarla çalışabilmeleri ve üretim alanında gezinebilmeleri gerekmektedir. Bu da onların çevrelerini kendi kendilerine algılayıp tanımlamalarını gerekli kılmakta, bu ise algılayıcılar sayesinde yapılmaktadır. Robota asıl can veren güç üstlendiği görevi komuta eden elektronik beynidir. Programlama ile oluşturulan uzman programlar bu alanın ürünlerindendir. Başta ABD'de olmak üzere ev işlerine yardımcı olan robotların kullanımı da giderek yaygınlaşmaktadır. Yerleri kendi kendine süpüren robot elektrik süpürgeleri büyük talep görmektedir. Bu nedenle bu alanda yapılan çalışmalar önemli birer kaynak teşkil etmektedir.

Çalışma alanındaki gezgin robot kısıtlı enerjisi ile kapsama işlevini gerçekleştirmek zorundadır. Görevini tamamlamadan enerjisi biten robotun yerini başka bir robot almakta, ya da robot, enerjisi azaldığında şarj noktasına gitmek zorunda kalmaktadır. Bu ise tekrarlı kapsamanın artması, etkinliğin azalması, kapsama için gerekli süre ve maliyetin artması demektir. Yine engelli ortamlarda karmaşıklık arttıkça robotun koordinasyon problemi yaşaması, hedefe yönelmesinde hata payını arttırır. Yapılan çalışmada da tam kapsamada en kısa yolu belirlemek adına yeni bir yaklaşım önerilmiş, bu sayede hem enerji tasarrufu sağlanmış hem de hata payı azaltılmıştır.

Tam kapsama problemleri, robotun işlevine göre, algılayıcı tabanlı ya da aparat tabanlı; robot sayısına göre, tek robot kapsama ya da çok robot kapsama; ortamın özelliğine göre, statik ortam ya da dinamik ortam olarak karşımıza çıkmaktadır. Araştırma kapsamında algılayıcı tabanlı gezgin çok robotlu statik çalışma alanının tam kapsanması için kapsamanın tamamlanma zamanını en küçükleyecek robot yollarının belirlenmesi probleminde yeni ve diğer yöntemlere göre avantajları söz konusu olan Çoklu Sarmal - Kapsar Ağaç Kapsama (ÇS-KAK) (MS-STC, Multiple Spiral - Spanning Tree Coverage) algoritması geliştirilmiştir. Önerilen algoritma bundan sonra ÇS-KAK olarak anılacaktır. Alanın çok robot tarafından kapsanması için, ele alınan yaklaşımda her robotun gezmekle yükümlü olduğu alanlarda ayrı kapsar ağaç oluşturularak robot yolu belirlenmiştir. İkinci bölümde literatürdeki gezgin tek robot ve çoklu robot kapsama algoritmaları tanıtılmıştır. Literatürdeki robot kapsama problemleri genel olarak (üretim açısından), hedef gözeten kapsama problemleri ve tam kapsama problemleri şeklinde sınıflandırılabilir. Hedef gözeten kapsama problemleri; montaj ve kaynak işleri yapan robotlar, bantlara malzeme besleyen robotlar, malzeme naklinde kullanılan robotlar gibi robotların bir noktadan bir hedefe gitmesini gerektiren problemlerdir. Tam kapsama problemleri ise; hijyenik ve anti bakteriyel üretim ortamlarının temizliği, ilaçlanması ve hijyen koşullarının denetimi, mayın ve maden tarama faaliyetleri, insan sağlığı için risk oluşturan ortamların (radyasyon, yangın, patlama vb.) izlenmesi ve denetimi gibi robotların belirli bir alanın tümünü taramasını gerektiren problemlerdir. Üçüncü bölümde ise, kapsama işlevinin kapsar ağaç yöntemi ile gerçekleştirildiği öne çıkan çalışmalar, alanda gelinen son noktayı da aktarmak ve kapsama mimarisini tanıtmak amacıyla detaylı olarak incelenmiştir. Dördüncü bölümde

çok robot tam kapsama üzerine önerilen yaklaşım tanıtılmıştır. Robot sayısının 10'a kadar çıkarılabildiği Visual Studio 2008 ortamında C# programlama dili ile geliştirilen program ara yüzü, elde edilen çıktılar ve önerilen algoritma adımlarını gösteren akış diyagramına yine bu bölümde yer verilmiştir. Beşinci bölümde, geliştirilen algoritma ve önceki çalışmalar karşılaştırılarak, algoritmanın diğer çalışmalara göre üstünlükleri, getirdiği avantajlar ve dezavantajlar farklı yerleşimler üzerinde çalıştırılarak örneklerle gösterilmiştir. Altıncı bölümde ise yapılan araştırmanın sonuçları ve gelecek çalışmalara ışık tutacak yönleri yer almaktadır.

Bu çalışmada önerilen gezgin robotların tam kapsama hedefine yönelik yol planlaması yaklaşımı özellikle modern üretim sistemleri gözetilerek geliştirilmiş olmasına karşın, sadece bu alanla sınırlı olmayıp tam kapsama hedefleyen tüm robot uygulamalarında kullanılabilecek özelliklere sahiptir.

2. ROBOT TAM KAPSAMASI ÜZERİNE YAPILMIŞ ÖNCEKİ ÇALIŞMALAR

Çok robot kapsar ağaç kapsama algoritması, gelişen teknoloji ile robot kullanım alanlarının artması sebebiyle son zamanlarda ilgi gören algoritmalar arasında yer almıştır. Oluşturulan yeni algoritmalar, yapılan iyileştirmeler, maliyetten ve zamandan tasarruf sağlayan geliştirmeler günümüzde daha önemli hale gelmiştir. Son yıllarda, bu amaca hizmet etmek için yapılan çoklu robot alanındaki çalışmalar dikkati çekmektedir.

2.1. Literatürdeki Gezgin Tek Robot Kapsama Algoritmaları

Literatürdeki gezgin tek robot kapsama algoritmaları; kapsar ağaç tabanlı kapsama, git-gel hücresel ayrıştırma tabanlı kapsama, üçgensel hücre ayrıştırma tabanlı kapsama, sezgisel kapsama algoritmaları olarak dört bölümde incelenebilmektedir. İlgili algoritmalar izleyen bölümde yer almaktadır.

2.1.1. Kapsar ağaç tabanlı kapsama algoritmaları

Sürekli alanda gezgin robotların kapsar ağacını oluşturma problemi Gabriely ve Rimon tarafından incelenmiştir. Statik Ortam KAK (Off-line STC), Dinamik Ortam KAK (On-line STC) ve Karıncamsı KAK (Ant-like STC) olmak üzere üç farklı algoritma ile kapsama analizi gerçekleştirilmiştir (Gabriely and Rimon, 2001a).

(Chang and Dan, 2006) çalışmasında dinamik ortam için kapsar ağaç oluşumu ele alınmış, (Gabriely and Rimon, 2001b) çalışmasından da esinlenilerek Dinamik Ortam Zigzag KAK (Zigzag On-line STC) ve Dinamik Ortam İlk En Büyük KAK (The Most Upper Side First On-line STC) algoritmaları incelenmiştir. Bu örneklerde yönlendirme (navigasyon) hatalarını en aza indirmek için çalışılmış, bunun ise daha az dönüş içeren algoritmalarla sağlanabileceği savunulmuştur.

Gezgin robotlar için sarmal yol kullanan kapsama stratejisi olan Geri dönüşlü Sarmal Algoritma (GSA, Backtracking Spiral Algorithm) Gonzalez vd. tarafından 2005 yılında geliştirilmiştir. Geri dönme mekanizması ile gezilmemiş bölgelerin tam kapsanması hedeflenmiştir. Tamamıyla boş hücreler değil, kısmi dolu olan hücreler de kapsanabilmek amacıyla algoritma genişletilmiş, diğer benzer algoritmalara da uygulanabilecek genel bir yöntem olarak geliştirilmiştir.

2.1.2. Git-Gel Hücresel Ayrıştırma tabanlı kapsama algoritmaları

Tam Hücresel Ayrıştırma yaklaşımı (Boustrophedon Cellular Decomposition) Choset tarafından geliştirilmiştir (Choset, 2000). Bu ayrışım çokgen engellere izin vererek tam kapsama gerçekleştirmektedir. Choset 2001 yılında kapsama alanı algoritmalarını, sezgisel, yaklaşık, kısmi-yaklaşık ve kesin hücresel bileşimler olarak dört bölümde incelemiştir.

Gezgin robotlarda tam kapsamayı gerçekleştirmek için yapılanmamış çevreyi kullanan ve bilinmeyen engellerden kaçınmaya yönelik oluşturulan algoritma Garcia ve Santos tarafından 2004 yılında geliştirilmiştir. Algoritma hücresel ayrışım metodu ile zigzag yaparak süpürme esasına dayanmaktadır

Huang (2001) tarafından maliyeti ve süreyi azaltan robot kapsaması oluşturma problemi ele alınmıştır. Kapsama bölgesini alt bölgelere ayırarak alt bölgeleri sırasıyla kapsayan, çokgen çevrelerde de uygulanabilen hat-süpürme tabanlı hücresel ayrıştırma algoritması (enküçük toplam irtifa (MSA- minimal sum of altitudes) ayrıştırması) geliştirilmiştir. Tam kapsamayı gerçekleştirmenin yanı sıra verimli yolu bulma problemi Yao tarafından ele alınmıştır (2006). Geleneksel süpürme hattı stratejisini kullanmanın sıklıkla tekrar taşınma gerektireceği, bunun ise düşük verimliliği doğuracağı savunulmuş ve gereksiz taşınmaları azaltan yeni süpürme hattı stratejisi sunulmuştur. Karmaşık çevrelerde robot yolunu oluşturan ve yeni süpürme hattı stratejisini kullanan bir geometrik algoritma tanımlanmıştır.

Kapsama alanı yol planlama algoritmalarında arka ve ön hareketli robotlar keskin dönüşleri almakta zorlanmaktadır. Önce yavaşlamalı, ters yöne yönelmeli ve sonra ters yönde ivmelenmelidir. Eğrisel yolu kullanarak kapsama zamanını enküçüklemeyi amaçlayan kapsar yol planlama algoritması (CPPA – coverage path planning algorithm) Surve vd. (2007) tarafından ele alınmıştır. Ancak bu yaklaşımın limiti oldukça büyük ortamda uygulanabilirliğidir.

(Kim, et al., 2007) çalışmasında alan kapsama görevini gerçekleştiren gezgin robotlar için, hücresel ayrışım tabanlı yeni bir yöntem önerilmiştir. Algoritmada verilen küçük hücreler büyük komşu hücreye birleştirilmiş ve böylece verilen çalışma alanı büyük birkaç hücreye ayrıştırılmıştır. Sonrasında her hücreye ön tanımlı şablon yolları atanmış ve bu yöntemle robot yolları belirlenmiştir.

2.1.3. Üçgensel Hücre Ayrıştırma tabanlı kapsama algoritmaları

(Oh, et al., 2003) çalışmasında temizleme robotları için tam kapsamayı gerçekleştirmede, çalışma alanını üçgensel hücre tabanlı ayrışıma tabi tutarak, üçgen hücreli harita üzerinde rota oluşturma problemi ele alınmıştır. Dikdörtgen hücreli haritada 8 rota yönü varken (çapraz hareketler de dahil edilmektedir), üçgen hücreli haritayla rota sayısı 12'ye çıkarılmış ve bu artış ile rota yolunun daha kısa ve esnek olması hedeflenmiştir.

2.1.4. Sezgisel kapsama algoritmaları

(Mei, et al., 2006) çalışmasında robotların kısıtlı enerjisinin tüketiminde verimliliğe dikkat çeken ve çalışma alanını keşfeden gezgin robotlar için enerji korunumlu yaklaşım önerilmiştir. Yaklaşımda öncelikle robot için sonraki hedef , sonrasında o anki pozisyon ve hedef arasında enerji korunumlu yol belirlenmiştir. Araştırmada tekrarlı kapsamaların azaltılması amaç edinilmiş, enerji tasarrufunun bu yolla da sağlanacağı vurgulanmıştır.

Araba benzeri araçlarla dinamik ortamda kapsama için izdüşüm eğrilerini en küçükleyen sarmal yol planlama algoritması Bosse vd. (2007) tarafından sunulmuştur. Bazı algılayıcı kümelerinin de kullanıldığı bu yöntem ile %95 oranında kapsama sağlanmıştır.

Kablosuz algılayıcı ağları yerleşiminde, statik yol planlama problemi Huang ve Záruba (2007) tarafından çalışılmıştır. Algılayıcıları öntanımlı alanda düzgün olarak dağıtan bir model geliştirilmiştir. Algılayıcı düğümlerinin yerleşimi için, öntanımlı statik yolu takip eden robot kullanılmıştır. Statik yollar içerisinde iki yeni yol olan Daireler ve S-Eğrileri (Circles ve S-Curves) yolları önerilmiştir.

Yapısal olmayan çevreye uyabilecek otonom temizleme robotları için yol planı oluşturma yaklaşımı Liu ve Zhu tarafından ele alınmıştır. Çevrenin çeşitliliği ve robotların limitli idrakı problemi ele alınmış, rassal yol planlama ve yerel bitmiş kapsama alanı yol planlamayı birleştiren, kombine kapsama alanı yol planlama stratejisi geliştirilmiştir. (Liu and Zhu, 2008)

Algılayıcı – tabanlı kapsama problemi için yol düzleştirme stratejisi Yufka vd. (2009) tarafından önerilmiştir. Düz yol oluşturmak kinematik kısıtlar içerir. Düz yol düğümlerdeki durmaları aza indirerek, gereksiz hızlanma ve yavaşlama hareketlerini de elimine etmektedir. Bu yaklaşım ile robotun köşeleri dönmesini kolaylaştırmak ve kapsamanın tamamlanma zamanını düşürmek hedeflenmiştir.

2.2. Literatürdeki Gezgin Çok Robotlu Kapsama Algoritmaları

Literatürdeki gezgin çok robot kapsama algoritmaları; kapsar ağaç tabanlı kapsama, git-gel hücresel ayrıştırma tabanlı kapsama, sezgisel kapsama, voronoi diagramı tabanlı kapsama algoritmaları olarak izleyen bölümde yer almaktadır.

2.2.1. Kapsar ağaç tabanlı kapsama algoritmaları

Çoklu robotlar için kapsama alanında fazlalık, etkililik ve sağlamlık Hazon ve Kaminka (2008) tarafından araştırılmış, çeşitli algoritmalar sunularak ilgili algoritmalar performans karşılaştırmasına tabi tutulmuştur. Çalışmada enküçük kapsar ağaç yöntemi baz alınmıştır. Robot yolları belirlenirken tek robot için oluşturulan kapsar ağacın kopyası kullanılmış, robotlara başlangıç noktaları atamak suretiyle her robotun yolu tayin edilmiştir. (Agmon, et al., 2006) çalışmasında çoklu robotlar için alan kapsamasında, kapsama süresinin bulunan kapsar ağaca bağlı olduğu ve ağaç yapısının önemli olduğu belirtilmiş, çoklu robot kapsamasında verimliliğin robotların birbirine uzaklıklarının eşit olmasına bağlı olduğu savunulmuştur.

Dinamik ortamda çoklu robot kapsaması üzerine kapsar ağaç algoritması Hazon vd. (2006) tarafından ele alınmıştır. Algoritmada robotlar başlangıç pozisyonu bilgisine sahiptir. Robot yolu engellendiğinde sonraki boş hücreden devam etmekte bu esnada tekrarlı kapsama oluşmaktadır. İki robot aynı hücreyi kapsamak istediğinde de sorun oluşmaktadır.

2.2.2. Git-Gel Hücresel Ayrıştırma tabanlı kapsama algoritması

Çoklu robot sistemlerinde tam kapsamanın Git-Gel Hücresel Ayrışım (Boustrophedon Cellular Decomposition) ile gerçekleştirildiği algoritma Kong vd. (2006) tarafından çalışılmıştır. Kapsama süresince hiçbir robotun boş kalmaması hedeflenmiştir.

2.2.3. Sezgisel kapsama algoritmaları

Algılayıcı tabanlı gezgin robot kapsar yol planlama problemi için genetik algoritma Kapanoğlu vd. (2009) tarafından geliştirilmiştir. Çalışma alanı disklerle modellenmiş, her diskin merkezinden geçen enküçük maliyetli yolu bulmak

hedeflenmiştir. Robotun seçeceği dikdoğrusal yön, 8 farklı örüntü ile önceliklendirilerek, boş olan komşu hücre öncelik sırasına göre yol oluşturan genetik algoritma geliştirilmiştir. Çoklu robot için kapsama süresini en küçükleyen genetik algoritma (HOGA- Hierarchical Oriented Genetic Algorithm) Özkan vd. (2009) tarafından geliştirilmiştir. HOGA, önceki çalışmada ele alınan tekrarlı kapsamayı en aza indiren tek rotanın oluşturulduğu ilk faz ve rotayı robotlara paylaştıran ikinci faz olmak üzere, iki fazdan oluşmaktadır.

Örüntü tabanlı çoklu robot kapsamasında, kapsama zamanını en küçükleyen genetik algoritma, Kapanoğlu vd. tarafından geliştirilmiştir. Robot dönüşleri robotun yavaşlaması, dönmesi ve tekrar hızlanmasını gerektirdiğinden kapsama süresinin hesaplanmasında dönüşler ayrıca değerlendirilmektedir (Kapanoğlu vd., 2010).

2.2.4. Voronoi Diagramı tabanlı kapsama algoritması

Çoklu robot algılayıcı tabanlı kapsama problemi için dinamik yol planlama yaklaşımını Yazıcı vd. (2009) tarafından ele alınmıştır. Ortamının dar olduğu ve bilinmeyen bölümler içerdiği varsayılmıştır. Algılayıcı tabanlı kapsama için Genelleştirilmiş Voronoi Diagramı tabanlı ağ yapısı kullanılmıştır. Başlangıçta robotlar eşit enerjiye sahiptir ve aynı noktada bulunmaktadır. CARP (Capacitated Arc Routing Problem) yaklaşımını kullanarak başlangıç tam kapsama rotası oluşturulmuştur. Bilinmeyen bölümler için tekrar planlama, klasik CARP için geliştirilmiş Ulusoy algoritması kullanılarak yapılmıştır. Yöntem diğer robot uygulama problemleri için uyarlanabilmektedir. Algılayıcı – tabanlı kapsama probleminde Genelleştirilmiş Voronoi Diagramı tabanlı grafik modelinin benimsendiği, yay kümesine bağlı başlangıç tam kapsama rotasında Çinli Postacı Problemi (CPP) ve/veya Kırsal Postacı Probleminin kullanıldığı çalışma yine Yufka vd. (2009) tarafından gerçekleştirilmiştir. Başlangıç rotası oluşturulduktan sonra, rota yeniden düzenlenmiş ve Ulusoy algoritması kullanılarak robotlara paylaştırılmıştır.

3. ÇOK ROBOTLU TAM KAPSAMA YÖNTEMLERİ

3.1. Gezgin Robotlar ile Sürekli Alan Kapsamada Kapsar Ağaç

(Gabriely and Rimon, 2001a) çalışmasında tek robot için kapsar ağaç yöntemi ile kapsama gerçekleştirilmektedir. Kapsama işlemi için kullanılan düzlemsel şekilli araç gezgin robota bağlanmaktadır. *A, engeller ile sınırlı, sürekli düzlemsel çalışma alanıdır*. Araç A'nın her noktasına taşınarak tam kapsama gerçekleştirilmektedir. *Her nokta bir kere kaplanırken hücreler tarafından oluşan kapsayan ağacın takip edildiği algoritma KAK(Kapsar Ağaç Kapsaması, Spanning Tree Coverage) olarak adlandırılır*.

Araç DxD boyutlu kare şeklindedir. Robotun aracı dikey yönde taşıyabildiği, (aracın 4 yanına doğru) aracı döndüremediği varsayılmaktadır. Çalışma alanı 2Dx2D boyutlu karesel hücrelere bölünmekte ve engellerle kısmi kaplanmış hücreler ihmal edilmektedir. Yaklaşımın kalitesi aracın büyülüğüne bağlıdır. D, çalışma alanı karakteristik boyutundan önemli ölçüde küçük düşünülmektedir. Çalışmada, Statik Ortam KAK (Off-line STC), Dinamik Ortam KAK (On-line STC) ve Karıncaması KAK (Ant-like STC) olmak üzere üç farklı algoritma ile kapsama analizi gerçekleştirilmiştir.

3.1.1. Statik ortamda kapsar ağaç kapsama algoritması

Statik Ortamda KAK'da robot, çevresi hakkında mükemmel öntanım bilgisine sahiptir. Algoritmanın girdisi çevrenin geometrik tanımı (sabit engeller, engelsiz sınırlı düzlemsel bölge geometrik tanımı) olarak düşünülmüştür.

Grafik yapısı G(V,E) ile tanımlanır. V; düğümdür, E'nin uç kısmı ve her hücrenin orta noktasıdır. E, komşu hücrelerin ortasını birleştiren doğru parçasıdır. Algoritmada engeller ile örtülü hücreler ihmal edilir. Ön işlemede S başlangıç hücresinden başlanır ve G için kapsar ağaç oluşturulur (herhangi bir kapsar ağaç

algoritması kullanılır). 2Dx2D boyutlu her hücre DXD boyutlu 4 hücreye bölünür. Kapsama işleminde; S'nin alt hücresinden başlanır, yol boyunca komşu alt hücreler arasında taşınılır (alt hücrelerin her biri araç şekline benzerdir). Taşımalar saat yönünün tersine kapsayan ağacın etrafı dolaşılarak yapılmalıdır. Başlangıç alt hücresine gelindiğinde kapsama işlemi sona ermektedir. Algoritmada robot kapsama aracını, aracın hareket yönüne bağlı olarak belirlenen, kapsar ağacın sağ tarafı üzerinde uzanan alt hücrelere doğru taşımaktadır. Araç bir noktaya kısıldığında ağacın etrafında dolaşılarak basit kapalı bir yol oluşturulur ve başlangıç alt hücresine geri dönülür.

3.1.2. Dinamik ortamda kapsar ağaç kapsama algoritması

Dinamik Ortamda KAK'de, robot çalışma alanı hakkında öntanım bilgisine sahip olmamakla birlikte, çalışma alanı kaplanırken çevrenin kapsar ağacını oluşturmada yerleşik algılayıcılar kullanılmaktadır. Robotun pozisyon ve yön bulma algılayıcıları bulunmaktadır. Böylece DxD boyutlu hücreleri tanıyabilmektedir. Menzil bulma algılayıcısı, o anki hücreye komşu dört hücredeki engelleri görmeyi sağlamaktadır. Kapsama aracı, çalışma alanının tamamını kaplayana kadar, artarak oluşturulan kapsar ağacın çevresini dolaşan alt hücre yolunu takip etmektedir.

Dinamik Ortamda KAK en uygun kapsama yolunu O(N) zamanda tamamlamakta, ziyaret edilmiş bölgeleri tanımlamak için O(N) belleğe ihtiyaç duymaktadır. Bu gereklilik çalışma alanı boyutunu sınırlı tutmaktadır. Algoritmada, robot saat yönündeki komşu hücreleri eşit olarak seçebilmekte olup, o durumda kapsama aracı saat yönünü baz alarak kapsar ağaç çevresini dolaşması gerekmektedir.

3.1.3. Karıncamsı kapsar ağaç kapsama algoritması

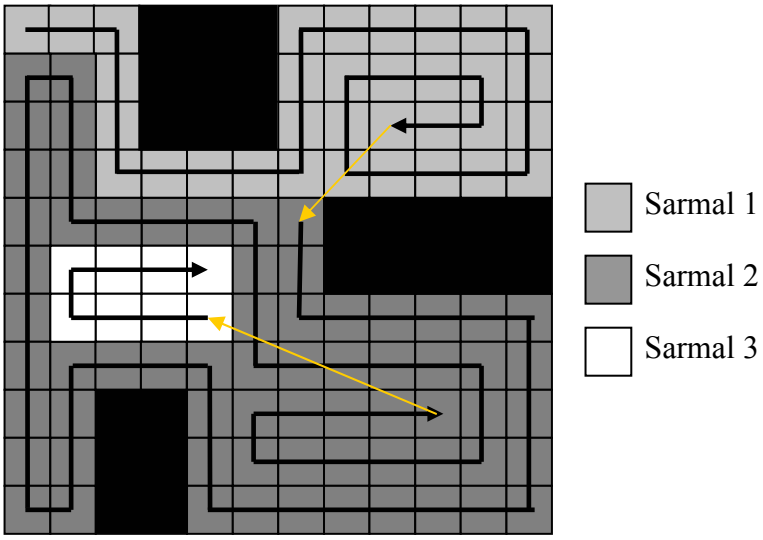
Karıncamsı KAK'da robot çevresi hakkında öntanım bilgisine sahip olmamakla birlikte, robot kapsama prosesi süresince feromon benzeri işaretleri ayırabilmektedir. Kapsama yolunu O(N) zamanda tamamlayabilmekte ve O(1) belleğe ihtiyaç

duymaktadır. Karıncamsı algoritma dinamik ortam algoritmasıdır ve Dinamik Ortam KAK ile aynı algılayıcıları (pozisyon, yön bulma, engel keşif, işaretçi keşif algılayıcıları) kullanmaktadır. Robot alt hücredeki renk, koku, sıcaklık veya pürüzlü yüzey gibi işaretleri ayırabilme yeteneğine sahiptir. İşaretçilerin ayrıntılı yapısı, robotların kaplanmış hücreleri tanımlamasına izin verdiği sürece önemli olmamaktadır. Komşu hücrenin 4 alt hücresi de işaretlenmemiş ise, bu hücre yeni komşuluk hücresi olarak belirlenmektedir. Robot tarafından saat yönünün tersi sırada yer alan hücre taranarak işaretlenmektedir.

Kapsar ağaç çalışma alanında yer alan kapalı eğridir. Robot bu ağacın çevresinde dolaşmaktadır. Kapsar ağaç bir hücreden iki kez geçememektedir. Bu nedenle Statik Ortam, Dinamik Ortam, Karıncamsı KAK algoritmaları tekrarlamasız kapsamadır. Üç KAK algoritması S'den her hücreye ulaşılabilen nitelikte kapsar ağaç üretmektedir. Dolayısıyla üç KAK algoritması başlangıç hücresi S'den ulaşılabilen her hücreyi kaplar.

3.2. Geri Dönüşlü Sarmal Tam Kapsama Algoritması

(Gonzáles, et al., 2005) çalışmasında, gezgin tek robot için, statik çalışma alanında, sarmal yol kullanan kapsama stratejisi olan Geri Dönüşlü Sarmal Algoritma (GSA, Backtracking Spiral Algorithm) önerilmiştir. Geri dönme mekanizması ile gezilmemiş bölgelerin kapsanması sağlanır. İri grid yapısı ile modellenmiş Basit GSA, kısmi kaplı hücreleri de kapsayabilmek amacıyla genişletilmiştir.



Şekil 3.1 Basit GSA algoritması

Basit GSA algoritmasında yüzey iri grid yapısı ile modellenir, her kare hücrenin boyutu robot boyutuna eşittir ve algoritma yalnızca tamamıyla boş hücreleri kapsar. İnce grid yapısında robot engelin yakınlarından geçebileceğinden daha iyi kapsama sağlar. Başlangıç olarak tüm hücreler bilinmez olarak işaretlenir, robot bir hücreyi kapsadığında sanal engel olarak işaretlenir. Engel içeren hücreler ise gerçek engel olarak işaretlenir.

Robot ileri hareket etmeden önce, Basit GSA birden fazla bilinmeyen hücre olup olmadığını değerlendirir. Yani alternatif muhtemel yol olup olmadığına bakılır *ve bu hücreler potansiyel Geri Dönen Noktalar (GDN, Backtracking Point) olarak işaretlenir*. Robot sarmal yolun bitiş noktasına ulaştığında geri dönme mekanizması çağrılır. İlk kaydedilen potansiyel GDN değerlendirilir, bu GDN seçildiğinden elenir. Seçimde GDN ve robot arasındaki öklit uzaklık kriter olarak kullanılabilir. Sanal engel olarak işaretlenmiş gezilmiş hücrelerden geçilerek seçilen GDN'ye ulaşılır. Seçilen GDN başlangıç noktası olmak üzere yeni sarmal oluşturulur. Aday GDN kalmadığında GSA durur çünkü gezilecek alternatif yol kalmamıştır, tüm boş hücreler gezilmiş olur.

Genişletilmiş GSA ile kısmi kapsanmış hücreler duvar takip prosedürü ile kapsanır. Genişletilmiş GSA'da sarmal yol oluştururken engel bulunduğunda duvar takip prosedürü çalıştırılır. Engelin çevresi gezilirken de alternatif yollar taranır. Engelin çevresi işaretler sayesinde tam olarak gezildiği tespit edildikten sonra sarmal yol oluşturmaya devam edilir. Bu yaklaşımın ana dezavantajı duvar takip ve genel kapsama algoritması ile oluşan yolların eşleşmeyebilmesidir.

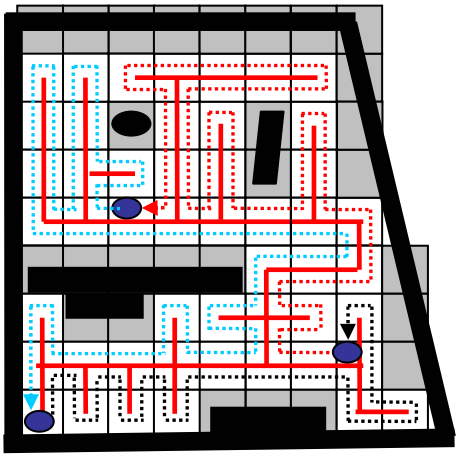
3.3. Çoklu Robotlar İçin Kapsamada Fazlalık, Etkinlik, Sağlamlık

(Hazon and Kaminka, 2008) çalışmasında çoklu robot kapsamasında sağlamlığa etkinliğe hitap eden Geri Dönüşsüz Çoklu Kapsar Ağaç Kapsama (Geri Dönüşsüz ÇKAK, Non backtracking Multiple Spanning Tree Coverage) algoritması ve Geri Dönüşlü Çoklu Kapsar Ağaç Kapsama (Geri Dönüşlü ÇKAK, Backtracking Multiple Spanning Tree Coverage) algoritmaları sunulmuştur. Robotun başlangıç pozisyonu algoritmanın kapsama süresini etkilemektedir. Geri dönüşlü algoritmada, robot bir hücreyi iki kez dolaşabilmektedir.

Çalışmadaki varsayımlara göre robot aracı DxD boyutundadır ve çalışma alanı 2Dx2D boyutlu hücrelerden oluşmaktadır. Robot algılayıcılar ve hareketi geçirici aparat gibi donanımlara sahiptir. Statik ortam çalışma alanı söz konusudur.

3.3.1. Geri dönüşsüz çoklu kapsar ağaç kapsama algoritması

Kapsama işi iki aşamada yapılır; birinci aşamada kapsar ağaç yolu oluşturulur ve bölümlere ayrılır, ikinci aşamada her robot KAK yolunun bir kopyasını kullanarak kendi bölümünü kaplar. Robotlar alt yollarını tamamladıklarında tüm alan kapsanmış olur. Eğer bir robot başarısız olursa, arkasındaki robot onun sorumluluğunu alır.



Şekil 3.2 Üç robot için kapsar ağaç

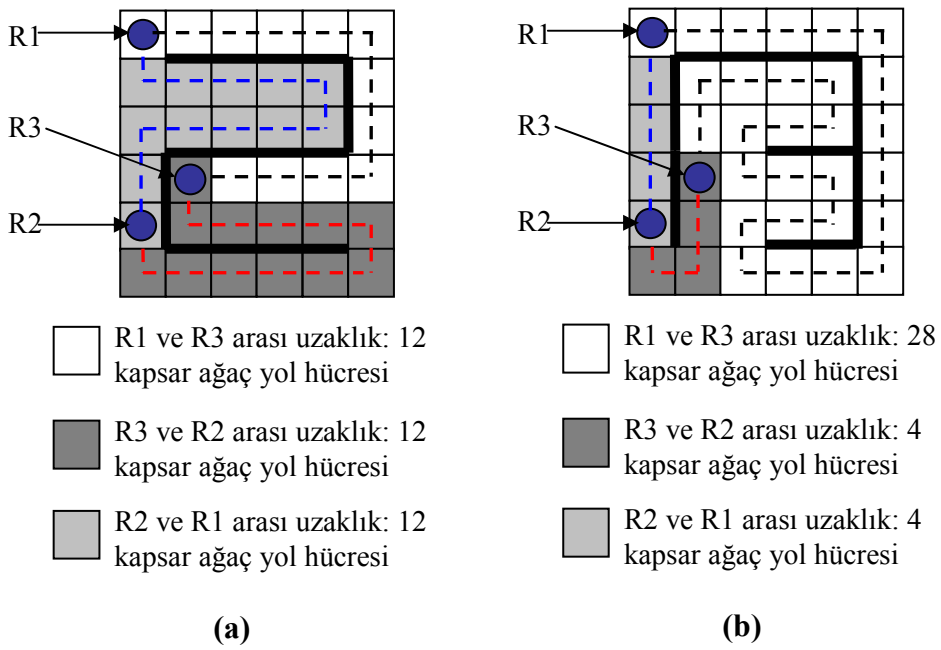
3.3.2. Geri dönüşlü çoklu kapsar ağaç kapsama algoritması

Geri Dönüşsüz ÇKAK'da iki robot ve aralarında tek alt hücre var ise robotlardan biri tek alt hücreyi, diğeri kalan n-3 hücreyi kapsayacaktır. Ancak robota geri dönme sağlanırsa tek hücreyi kapsayan robot kapsama sonrası diğer yöne gidebilir ve iki robot n-3 bölümün ortasında karşılaşır. Geri dönüş ile robotların kat edecekleri mesafeler dengelenmeye, başlangıç noktalarındaki uygunsuzluklar giderilmeye çalışılmaktadır.

3.4. Etkili Çoklu Robot Kapsaması İçin Kapsar Ağaç Yapılandırması

(Agmon, et al., 2006) çalışmasında ağaç yapısının önemini vurgulayan, her farklı kapsar ağacın farklı kapsama süresi oluşturduğu çoklu robot kapsama algoritması önerilmiştir. Algoritmadaki kapsama verimliliği, robotların birbirlerine olan uzaklıklarının eşit olmasına bağlıdır.

Çalışma alanı statik yapıda olup 2Dx2D boyutlu kare hücrelerden oluşmaktadır. Robota ise DxD boyutunda aparat bağlıdır ve robot dikey yönde hareket edebilmektedir.



Şekil 3.3 Farklı ağaçların kapsama süresine etkisi

Şekil 3.3'te N=36 hücre ve k=3 robot bulunmaktadır. A ve b seçeneklerinde, robotların başlangıç pozisyonları aynı olmasına rağmen, farklı ağaç yapısı robotlara düşen alanı ve dolayısıyla kapsama süresini etkilemektedir.

Çalışmada robot başlangıç pozisyonları verildiğinde, [N/k] uzaklığını sağlamaya yönelik ağaç yaratma prosedürü sunulmuştur. Prosedür, başlangıç pozisyonu verildiğinde, robotlar için alt ağaç oluşturur. Hücreler diğer ağaçlardan uzaklığı çoğaltan yolda seçilir. Tüm k alt yolları oluşturulduktan sonra, bu yollar birleştirilir. Alt yollar birleştirilerek, robotlar arası mesafenin [N/k] olduğu, tek bir kapsar ağaç yolu oluşturmak hedeflenir.

3.5. Çoklu Robot Kapsar Yol Planlamasına Genetik Algoritma Yaklaşımı

İki fazdan oluşan HOGA (Hierarchical Oriented Genetic Algorithm) Özkan vd. tarafından 2009 yılında önerilmiştir. İlk faz - OGA (Oriented Genetic Algorithm), önceki araştırmalarında da (Kapanoğlu vd., 2009) detaylı bir şekilde yer verdikleri tek robot için tekrarlı kapsamayı en aza indirmeyi amaç edinen rotayı oluşturmakta; ikinci faz - DGA (Directed Genetic Algorithm) oluşturulan yolu robotlara paylaştırmaktadır.

Çalışma alanı statik yapıda olup D çaplı disklerden oluşmaktadır. Robot, çalışma alanı diskleriyle aynı boyutta olup, dikey yönde hareket etmektedir.

OGA ile tüm diskleri en az bir kere gezerek kapsama süresini en küçükleyen dik doğrusal hareketler içeren bir yol tanımlanmıştır. Robot hareketlerine örüntü ile belirtilen komşu disk öncelik sırası rehberlik etmektedir. Şekil 3.4'de algoritmada kullanılan sekiz farklı örüntüden ikisi yer almaktadır. Robot C öneki ile belirtilen örüntünün orta noktasında yer almaktadır. 1 numaralı disk eğer gezilmemişse ve boşsa en yüksek önceliğe sahiptir. Örüntü, çıkmaz noktaya ulaşılana kadar, robota rehberlik eder. Çıkmaz noktaya ulaşıldığında en az bir disk tekrar kapsanarak en kısa mesafedeki komşu diske gidilir ve yol oluşturmaya devam edilir.

			13			
		24	5	14		
	23	12	1	6	15	
22	11	4	C1	2	7	16
	21	10	3	8	17	
		20	9	18		
			19			

			19			
		20	9	18		
	21	10	3	8	17	
22	11	4	C2	2	7	16
	23	12	1	6	15	
		24	5	14		
			13			

Şekil 3.4 Komşu-disk önceliklendirme örüntüsü

Önerilen algoritmadaki her gen örüntü tarafından belirtilen, sonraki ziyaret edilecek komşuluk numarasını gösterir. N-disk kapsama problemi için kromozom (N-1) gen içerir. (Şekil 3.5)

Kromozom 1

Gen	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Komşu	2	2	2	2	2	3	3	3	3	4	1	1	1	4	4	11	3	3	2	2	2	3	4	4	4

Şekil 3.5 OGA için kromozom yapısı

OGA'nın adımları;
Başlangıç popülasyonu oluştur,
Do
    Değerlendirme & seçim
    Yeniden oluşturma
    Çaprazlama
    Mutasyon
    Genişletilmiş yeni popülasyon
While (durma kriteri sağlanmamış)

Birinci fazda oluşturulan yol, DGA ile robotlara paylaştırılır ve her robot için yol başlangıç ve bitiş noktaları belirlenir. DGA adımları OGA ile aynıdır. DGA kromozomları iki bölümden oluşur. Birinci bölümde robotların gezeceği disk sayısı, ikinci bölümde hareket yönü belirlenir. M robot için gerekli gen sayısı 2M dir.

10	20	10	0	1	1

Şekil 3.6 DGA için kromozom yapısı

Şekil 3.6'da üç robot ve 40 – disk problemi için örnek kromozom yapısı gösterilmiştir. Robotlara sırasıyla 10, 20 ve 10 disk yüklenmiştir. İlk robot disk 1'den başlar ve disk 10'da yolunu tamamlar. İkinci robot 30. diskten 11. diske, üçüncü robot ise 40. diskten 31. diske olan bölümü kapsar. Bu gösterim ile robotlar için uygun yön ve alt yol belirlemek hedeflenmiştir.

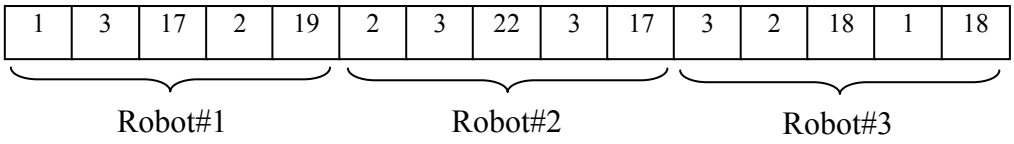
3.6. Çoklu Robot Kapsar Yol Planlamasında Kapsama Süresini En Küçükleyen Genetik Algoritma Yaklaşımı

Örüntü tabanlı çoklu robot kapsamasında, kapsama zamanını en küçükleme maksatlı dönüş sürelerini de dikkate alan yeni bir genetik algoritma, Kapanoğlu vd. (2010) tarafından geliştirilmiştir. Çalışma alanı statik yapıda olup D çaplı disklerden oluşmaktadır. Robot, çalışma alanı diskleriyle aynı boyutta olup, dikey yönde hareket etmektedir.

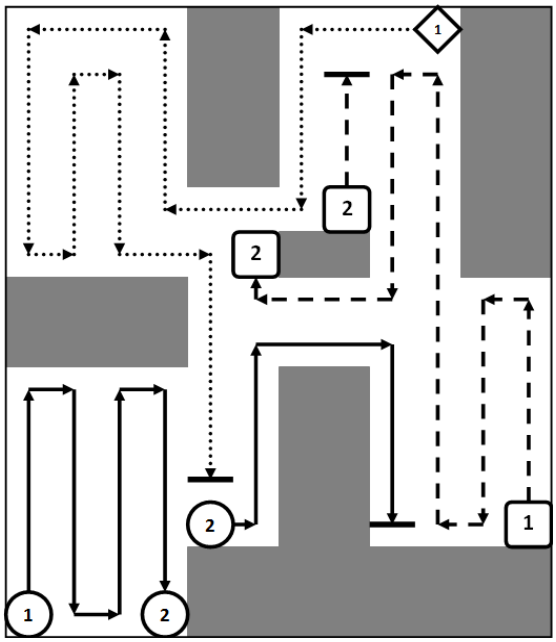
Robot hareketlerine sekiz farklı örüntüde belirtilen komşu disk öncelik sırası rehberlik etmektedir. Robot örüntünün orta noktasında yer almaktadır. 1 numaralı disk eğer gezilmemişse ve boşsa en yüksek önceliğe sahiptir. Örüntü, çıkmaz noktaya ulaşılana kadar, robota rehberlik eder. Çıkmaz noktaya ulaşıldığında en az bir disk tekrar kapsanarak en kısa mesafedeki komşu diske gidilir ve yol oluşturmaya devam edilir.

Önerilen algoritmadaki kromozom yapısında her robot için üç parametre bulunmaktadır. Birinci parametre robotun öncelik sırasını, ikinci parametre robot için seçilen örüntü numarasını, üçüncü parametre o örüntüye bağlı gezilecek disk sayısını belirler. Bir robot için uygulanabilecek örüntü sayısı birden fazla olabilir ve kromozom

uzunluğunda robot sayısı ile birlikte belirleyicidir. 3 robot ve 15 genli bir kromozom için ilk 5 gen birinci robotun parametrelerini, ikinci 5 gen ikinci robotun parametrelerini ve üçüncü 5 gen üçüncü robotun parametrelerini gösterir. Her bölümün ilk geni robotun öncelik sırasını, bölümdeki takip eden genler eşli olarak robotun takip edeceği örüntü numarasını ve kapsayacağı disk sayısını belirtir.



(a)



(b)

Şekil 3.7 (a) Örnek kromozom (b) Üç robot ve izin verilen örüntü sayısı iki iken elde edilen çözüm

Şekil 3.7 (a) seçeneğinde ilk robot 1. gende yer aldığı üzere birinci önceliğe sahiptir. 2. gen ilk robotun ilk olarak 3 numaralı örüntüyü takip edeceğini, 3. gen bu örüntüye bağlı olarak 17 diski kapsayacağını, 4. ve 5. genler ilk robotun sonrasında, 2

numaralı örüntüyü takip ederek 19 diski kapsayacağını göstermektedir. Kromozomun takip eden genlerinde ikinci ve üçüncü robota ait parametreler yer almaktadır. Şekil 3.7 (b) seçeneğinde üç robot daire, kare ve karo şekillerle betimlenmiştir. Numara 1 robotların başlangıç pozisyonlarını, Numara 2 ve daha büyük numara ile belirtilmiş şekiller tekrarlı kapsamaların oluştuğu yolların başlangıç ve bitim noktalarını göstermektedir.

Algoritmanın adımları;
Başlangıç popülasyonu oluştur,
While (durma kriteri sağlanmamış)
    Değerlendirme & seçim
    Yeniden oluşturma
    Çaprazlama
    Mutasyon
    Genişletilmiş yeni popülasyon
Loop

Başlangıç popülasyonu oluşturmada, popülasyon büyüklüğü, robotun takip edeceği örüntü sayısı, her robot için hareket sayısı kullanıcı kontrollü olarak belirlenir. Fakat her robot için hareket sayısı (toplam disk sayısı/robot sayısı)±10% oranını geçemez. Sonraki iterasyonlarda bu kısıt gözetilmez. Robot öncelikleri 1'den robot sayısına kadar olan numaralar içinden tek olarak rassal seçilir.

Değerlendirme ve seçimde uyumluluk değeri toplam veya en küçük/en büyük tamamlanma zamanını belirtir. Her bir robot için tamamlanma zamanı;

Düz hareket sayısı*t_s + dik açıda dönüş sayısı*t_{ra} + 180ºde dönüş sayısı*2*t_{ra}

Amaç en büyük tamamlanma zamanını en küçüklemektir. Yeniden oluşturmada elitizm ile stokastik turnuva seçimi uygulanmıştır (Kapanoğlu vd., 2010).

Çaprazlama operasyonunda, tek nokta çaprazlama kullanılmış takibinde düzeltme işlemine tabi tutulmuştur. Robot öncelik sırasında değişiklik söz konusu olduğunda, kromozom sağ tarafı düzeltme işlemine tabi tutulur. Robota atanan örüntü

numarasında oluşan değişiklik diğerlerinden bağımsız olduğundan düzeltme işlemi gerektirmez. Robotlar için oluşan toplam hareket sayısı toplam disk sayısından farklı olamaz. Eğer çaprazlama sonucu böylesi bir durum söz konusu olduysa her bir robota atanan hareket sayıları disk sayısına eşit olana kadar arttırılır veya azaltılır.

Mutasyon işleminde, rassal seçilen genin değeri uygulanabilir rastgele bir değerine değiştirilir. Mutasyon robot öncelik sırasını değiştirmişse, daha önce o önceliğe sahip robot geni ile yer değiştirme uygulanır. Robota atanan örüntü numarasında mutasyon genin değerini diğer genlerden bağımsız olarak 1 ile 8 arasında bir değere değiştirir. Robotun örüntüye bağlı hareket sayısında oluşan mutasyon ile hareket sayısı 1 ile toplam disk sayısı arasında bir değere değiştirilir. Kromozomdaki toplam hareket sayısının sabit kalması için her robota atana hareket sayılarında düzeltme işlemi yapılır.

4. ÇOK ROBOTLU TAM KAPSAMA PROBLEMİ İÇİN GELİŞTİRİLEN YAKLAŞIM

Çoklu robotlar için enküçük kapsar ağaç tam kapsama algoritması üzerine, literatürde geçen çalışmalara bir önceki bölümde detaylı bir şekilde yer verilmiştir.

Kapsar yol planlaması enküçük kapsar ağaç kullanılarak, robotun tarama alanındaki her noktaya ulaşabilmesi için yolun belirlenmesidir. Kapsar ağaç, bir serim (graf) üzerindeki tüm düğümleri içeren ve (düğüm_sayısı –1) kenardan oluşan alt serimdir. Tüm düğümleri içermesi nedeniyle kapsar ağaç olarak adlandırılır. Kenarlar çift yönlü bağlantıları gösterdiği ve yapı halka içermediği için bu ağaçlarda herhangi iki düğüm arasında sadece tek bir yol bulunur. Bir serim üzerinde en küçük kapsar ağacı tespit etmek için çeşitli algoritmalar mevcuttur. Bunlardan Kruskal ve Prim algoritması Ek-1'de yer almaktadır.

Kapsar ağaç yöntemi ile, statik ortamda, çoklu robotlar için tam kapsama gerçekleştiren algoritmalara göre daha kısa sürede kapsama sağlayarak; daha verimli, pratik yol oluşturan, alandaki çalışmalara büyük katkı sağlayan, çalışma kapsamında geliştirilen ÇS-KAK algoritması izleyen bölümde yer almaktadır.

4.1. Çoklu Sarmal - Kapsar Ağaç Kapsama Algoritması Tanımı

Önerilen algoritmada, çalışma alanı 2Dx2D boyutlu grid tipi hücrelerden oluşmaktadır. Robot ise DxD boyutlu olup kapsar ağacın çevresinden dolaşabilecek şekilde tasarlanmıştır. Çalışma alanındaki her hücre kapsar ağaçtaki bir düğümü belirtmektedir. Komşu hücreler karşı gelen hücre ile bir kenar oluşturur. Bu serim komşuluk serimi (adjacency graph) olarak adlandırılır. Gezgin satıcı probleminde olduğu gibi her düğüm en az bir kere ziyaret edildiğinde kapsama alanı robot tarafından kapsanmış olur. Komşu hücreler arası oluşturulan yolların ağırlıkları birbirine eşit ve 1

olarak kabul edilmiştir. İyi kapsama algoritmaları oluşturulan kapsama yolunu kısaltmakta ve her hücreden sadece bir kere geçilmesini sağlayarak tekrar eden kapsamaları en aza indirmektedir. Geliştirilen algoritmada da çoklu robotlar için tekrar eden kapsama oluşmaması için birbirine komşu olan hücreler arasında kapsama yolu oluşturulmuş ve mümkün olan kısa yol elde edilmiştir. Çoklu robot kullanımı ile kapsamanın tamamlanma zamanı kısalmakta, enerjisi biten robotun yerini bir diğeri alabilmektedir.

Bu çalışmada ele alınan problem, çalışma alanı hakkında öntanım bilgisine sahip iken (statik ortam, içinde engel olan hücreler ve taranması gereken boş hücreler konumu ve sayısı biliniyorken), alanın tam kapsanması için birden fazla robot kullanıldığında robotlara kapsayacakları alanı eşit düzeyde paylaştıran, alan bölünmesini en aza indiren enküçük kapsar ağaç yolu oluşturmaktır. ÇS-KAK algoritması, statik çalışma alanının çok robot tarafından sarmal kapsar ağaç yöntemiyle tam kapsanması esasına dayanır. Algoritmada öncelikle robot kapsama işlevini gerçekleştirecek robot sayısı belirlenir. Çalışma alanı 2Dx2D boyutlu grid tipi hücrelerden oluşmakta ve içinde engel olan hücreler dolu kabul edilmektedir. Robot sayısı belirlendikten sonra çalışma alanındaki toplam hücre sayısından dolu hücre (kısmi ya da tam olarak engel ile kaplı hücreler) sayısı çıkarılarak çalışma alanındaki toplam boş hücre (robot tarafından kapsanması gerekli hücreler) sayısı bulunur. Toplam boş hücre sayısı robot sayısına bölünerek bir robota düşen kapsanacak hücre sayısı belirlenmiş olur. Bu sayede her robota eşit iş yükü dağıtılmış olur. Aynı zamanda robotların birbirine yakın zaman aralıklarında kapsama işlevini gerçekleştirmesi sağlanarak, kapsamanın tamamlanma zamanının enküçüklenmesi sağlanmış olur. Bir robotun kapsayacağı hücre sayısı belirlendikten sonra, sıra hangi hücrenin hangi robot tarafından kapsanacağına gelmektedir. Alan bölünmesi oluşmaması için hücreler robotlara atanırken aynı sıradaki komşu hücrelere öncelik verilmektedir. Aynı sırada engel ile karşılaşıldığında ya da son hücreye ulaşıldığında alt sıradaki komşu boş hücre robota atanmaktadır. Bu sayede oluşabilecek alan bölünmelerinin oluşmaması, alan bölünmesinin kaçınılmaz olduğu durumlarda ise en az sayıda robot için oluşması sağlanmış olur. Atamaya alandaki her bir hücreden ayrı ayrı başlandığında oluşan, robotlara atanan hücre kombinasyonları elde edilir. Alanı robotlara paylaştıran hücre kombinasyonları değerlendirilerek en az alan

bölünmesinin, en az sayıda hücre için gerçeklendiği ve toplam tekrar kapsanan hücre sayısının en düşük olduğu atama seçilir. Her bir robot için kapsanacak hücreler belirlendikten sonra robotların kapsama yolları oluşturulur. Kapsama yolu oluşturmada sarmal kapsar ağaç kapsama algoritması kullanılmıştır. Algoritmanın en büyük avantajı başlangıç noktası açısından alternatif seçenekler sunmasıdır. Robotun kapsama işlevine başlayacağı hücre olan başlangıç noktası, robota atanan hücrelerden herhangi biri olabilir ve bu çözümün en iyiliğini değiştirmez. Robot kendi alanı içindeki hangi hücreden başlarsa başlasın, kapsama işlevi bittiğinde başladığı hücreye dönmüş olacaktır.

Sarmal kapsar ağaç yol oluşturma algoritmasında, saat yönü veya saat yönünün tersi seçilerek robot hareketleri yönlendirilir. Hangi yönün seçileceği çözümün en iyiliğini değiştirmez ancak yön saat yönü veya saat yönünün tersi şeklinde başta belirlenir ve algoritma süresince değiştirilmez. Algoritmada saat yönünün tersinde robot hareketlerinin belirlendiğini düşünürsek, sırasıyla robot sağ tarafına, sağdaki hücre daha önce kapsanmış ise veya engel ile kaplıysa öne, öndeki hücre daha önce kapsanmış veya engel ile doluysa sol tarafına yönlendirilir ve dolayısıyla kapsar ağaç o hücreye bağlanır. Robotun kapsayacağı son hücreye ulaşılana kadar bu işleme devam edilir. Sol taraftaki hücrede dolu veya daha önce kapsanmış ise kapsar ağaç takip edilerek (gerekirse kapsar ağaç etrafından dolaşılarak), kapsar ağaca komşu kapsanmamış ve boş robota ait hücre taranır. Tarama işlemine kapsanacak komşu hücre bulunana veya başlangıç noktasına ulaşılıncaya kadar devam edilir. Komşu hücre bulunduğunda kapsar ağaç hücreye bağlanır ve sağ,ön,sol sırasıyla boş ve kapsanmamış hücre araştırma işlemine devam edilir. Kapsar ağaç takip edilirken başlangıç noktasına ulaşılmışsa, robot için alan bölünmesi oluşmuş demektir. Bu durumda, robota ait diğer alanda kapsar ağaç yolu oluşturmaya devam edilir. Tüm robotlar için kapsar ağaç kapsama yolu oluşturulduktan sonra, robotlar başlangıç pozisyonlarına getirilir ve ağaç solda kalacak şekilde robot tarafından yol takip edilir. Algoritma adımları ve akış diyagramları izleyen kesimde yer almaktadır.

4.2. Çoklu Sarmal-Kapsar Ağaç Kapsama Algoritma Adımları ve Akış Diyagramı

Algoritma 1 - Sarmal kapsar ağaç yol oluşturma algoritması;

1. IF Robot sağındaki komşu hücre boş ve kapsanmamış; THEN kapsar ağacı sağdaki hücreye bağla, adım1'e git.

2. IF Robot önündeki komşu hücre boş ve kapsanmamış ise; kapsar ağacı öndeki hücreye bağla, adım1'e git.

3. IF Robot solundaki komşu hücre boş ve kapsanmamış ise; kapsar ağacı soldaki hücreye bağla, adım1'e git.

4. IF Robot için son hücrede bulunuluyor ise; adım 9'a git.

5. Robota ait boş ve kapsanmamış komşu hücre bulmak için kapsar ağaç yolunu (gerekirse etrafından dolaşarak) takip et.

6. IF Robota ait boş ve kapsanmamış hücre bulundu ise; kapsar ağacı komşu hücreye bağla ve Adım 1'e git.

7. ELSE IF Başlangıç hücresine ulaşılmadı ise; Adım 5'e git.

8. ELSE Alan bölünmesi; Robota ait boş ve kapsanmamış ilk hücreye taşın ve Adım 1'e git.

9. Son.

Algoritma 2 – Çoklu robotlar için enküçük kapsar ağaç tam kapsama algoritması;

1. Robot sayısını (R) belirle.

2. Çalışma alanındaki engelli hücre toplamını (E) bul.

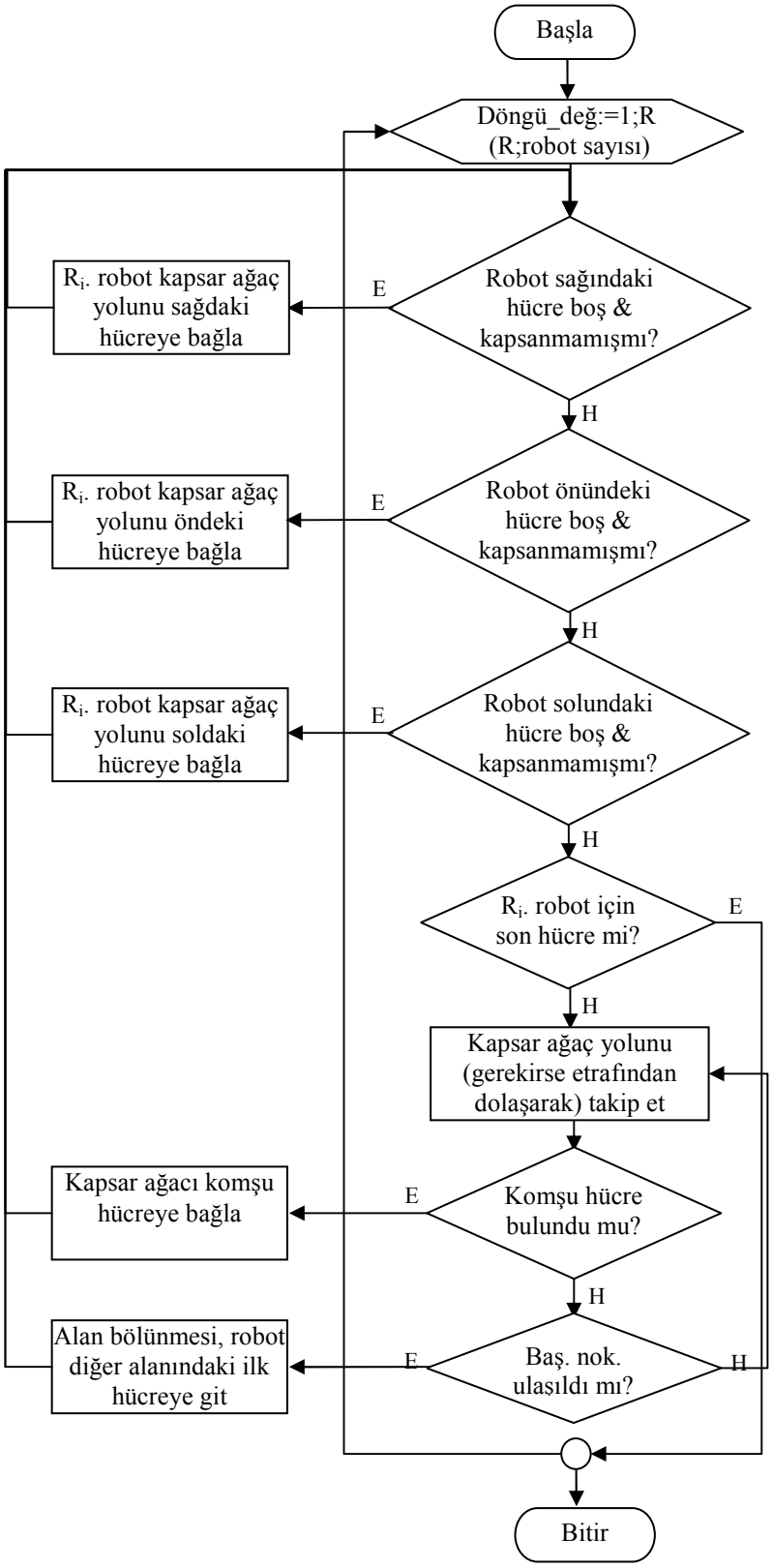
3. Bir robot tarafından kapsanacak hücre sayısını bul ((T-E)/R; T:Toplam hücre sayısı).

4. Tekrar kapsamayı en düşükleyen hücresel kombinasyonu bul

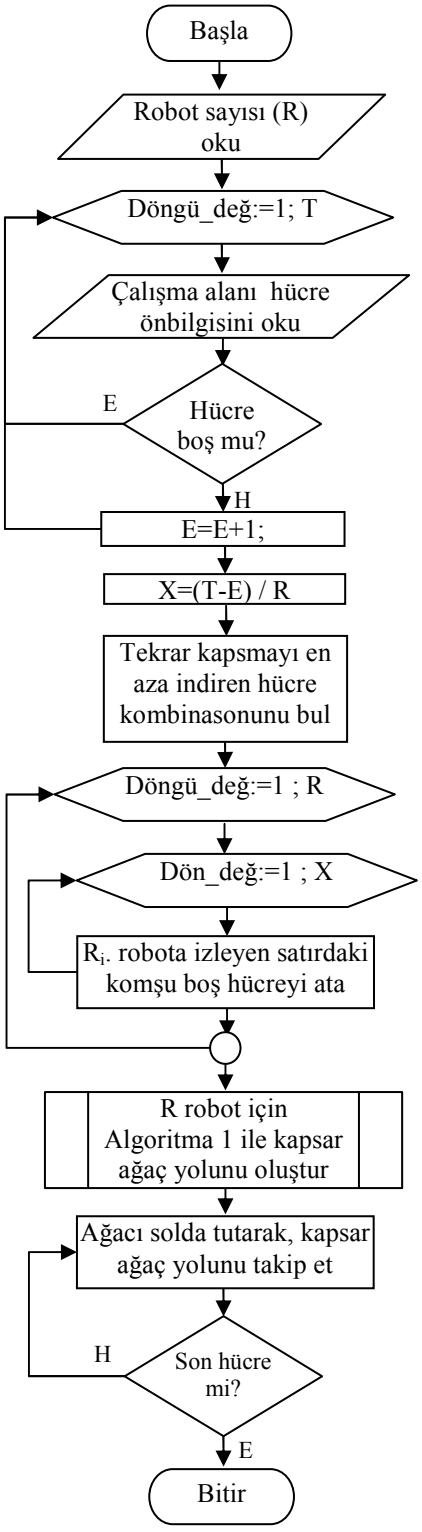
5. Robotlara kapsayacakları hücreleri (4. adımda bulunan kombinasyona göre) ata.

6. Her bir robot için Algoritma 1 ile kapsar ağaç kapsama yolunu oluştur.

7. Yol solda kalacak şekilde, robot kapsar ağaç yolunu takip et.



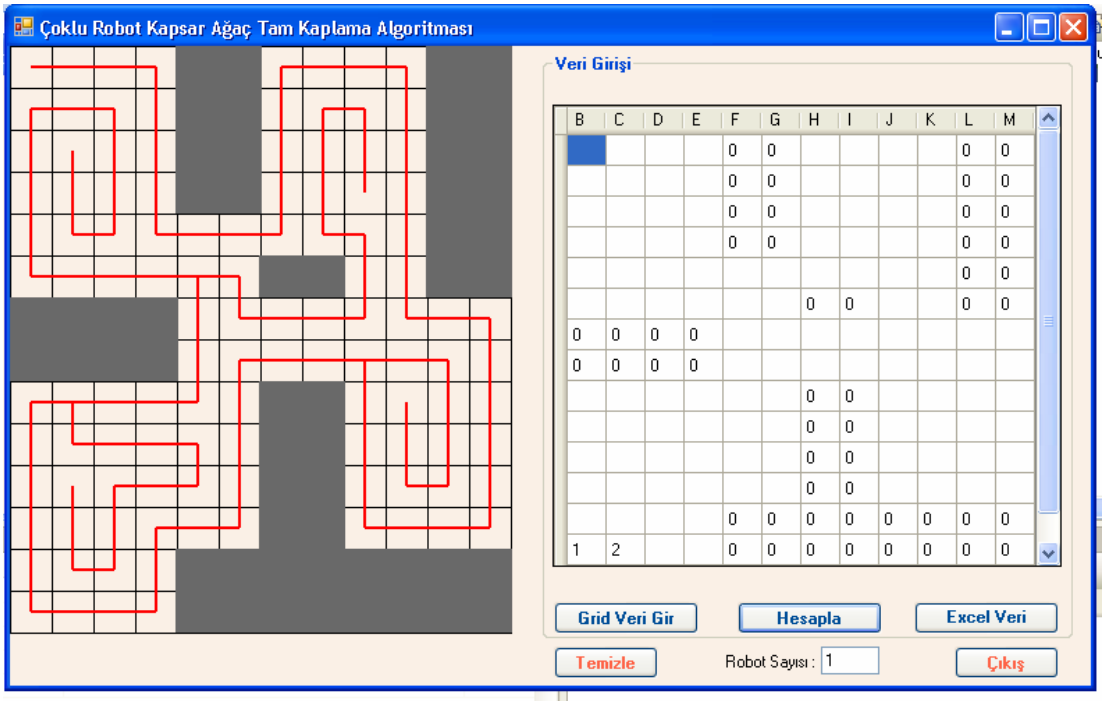
Şekil 4.1 Algoritma 1, R robot için sarmal kapsar ağaç yol oluşturma algoritması akış diyagramı



Şekil 4.2 Algoritma 2, çoklu robotlar için enküçük kapsar ağaç tam kapsama algoritması akış diyagramı

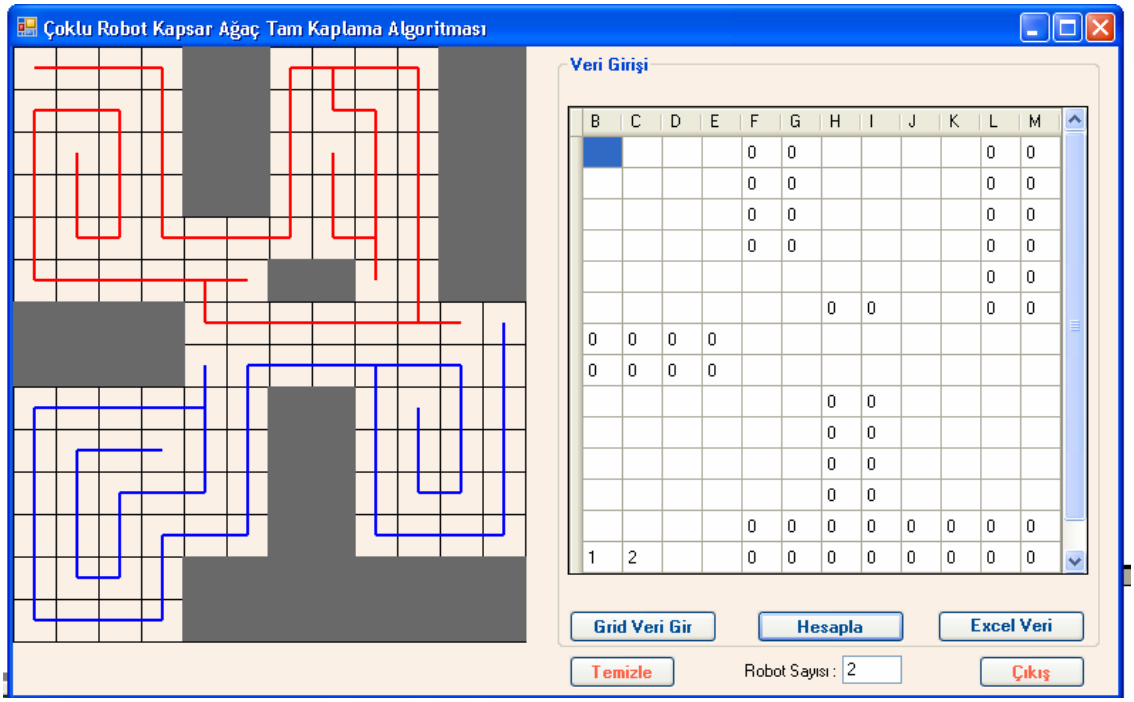
4.3. Çoklu Sarmal - Kapsar Ağaç Kapsama Algoritması C# Programı

Algoritma Visual Studio 2008 ortamında C# dili ile programlanmıştır. Excel'den veri yükleme ya da gridden hücre ön bilgisini girme yoluyla çalışma alanı bilgisi programa yüklenir, robot sayısı da belirtildikten sonra hesapla butonuna basılarak kapsar ağaç yolu kolayca hesaplanır. Şekil 4.3'te C# programı ile tek robot için kapsar ağaç yolu hesaplanmıştır.



Şekil 4.3 ÇS-KAK algoritması C# Programı

Sarmal kapsar ağaç tam kapsama algoritmasında, başlangıç noktası robot kapsar ağacı üzerindeki herhangi bir hücre olabilmektedir. Hangi hücreden başlanırsa başlansın kapsama sonunda başlanılan noktaya geri dönülecektir. Çünkü kapsar ağaç ile basit kapalı bir yol oluşmaktadır.



Şekil 4.4 İki robot için ÇS-KAK algoritması C# Programı

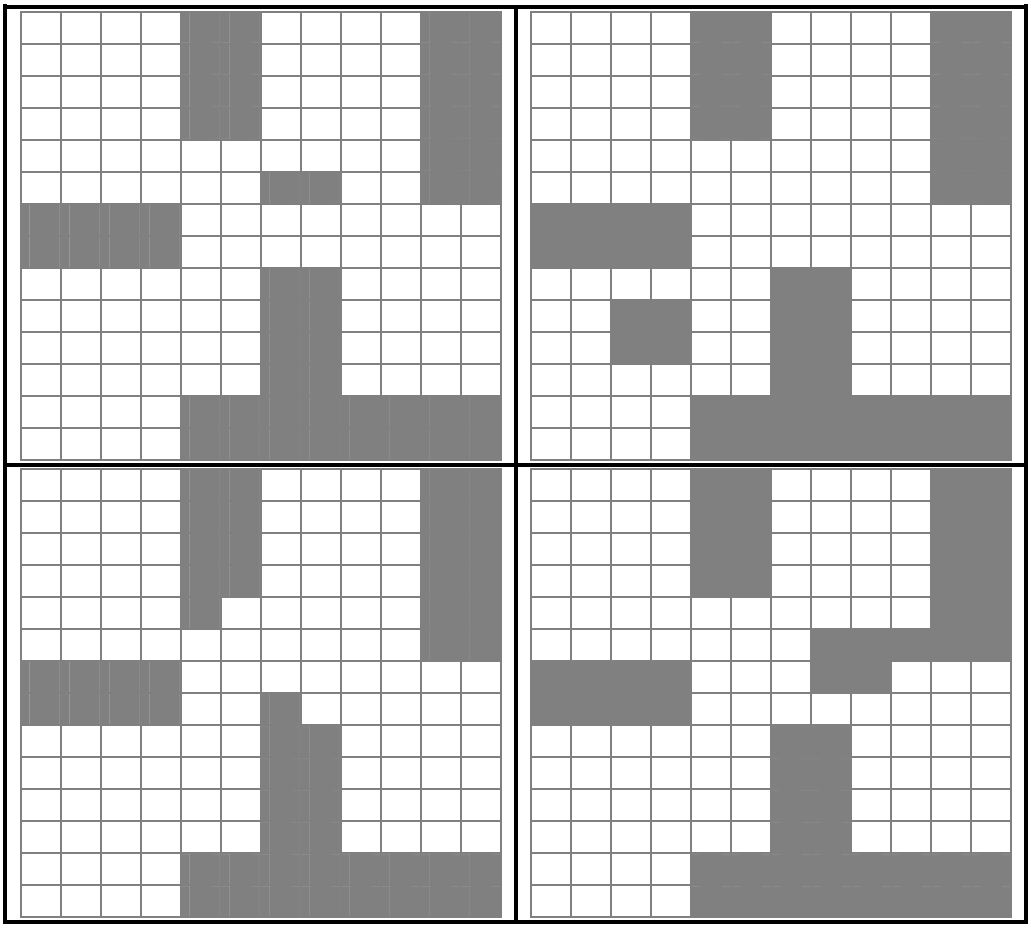
Tam kapsamanın sağlandığı algoritmada, her bir robota eşit sayıda hücre ataması yapıldığından, her bir robota eşit iş yükü de paylaştırılmakta, bu sayede kapsama süresinde de iyileşme sağlanmaktadır. Şekil 4.4'te de görüldüğü üzere, 57 boş hücre kırmızı yolla gösterilen birinci robota atanmış iken, diğer 57 boş hücre mavi yolla gösterilen ikinci robota atanmıştır.

Aynı zamanda geliştirilen algoritma Visual Studio 2008 ortamında C# dili kullanılarak kodlanmış olup çözüm, saniyeler süresinde elde edilebilmektedir. (İlgili kodlar Ek.2'de yer almaktadır.) Bu ise alanda gerçekleştirilen diğer çalışmalara göre büyük bir avantaj sağlamaktadır.

4.4. Önerilen Yaklaşımın Testi

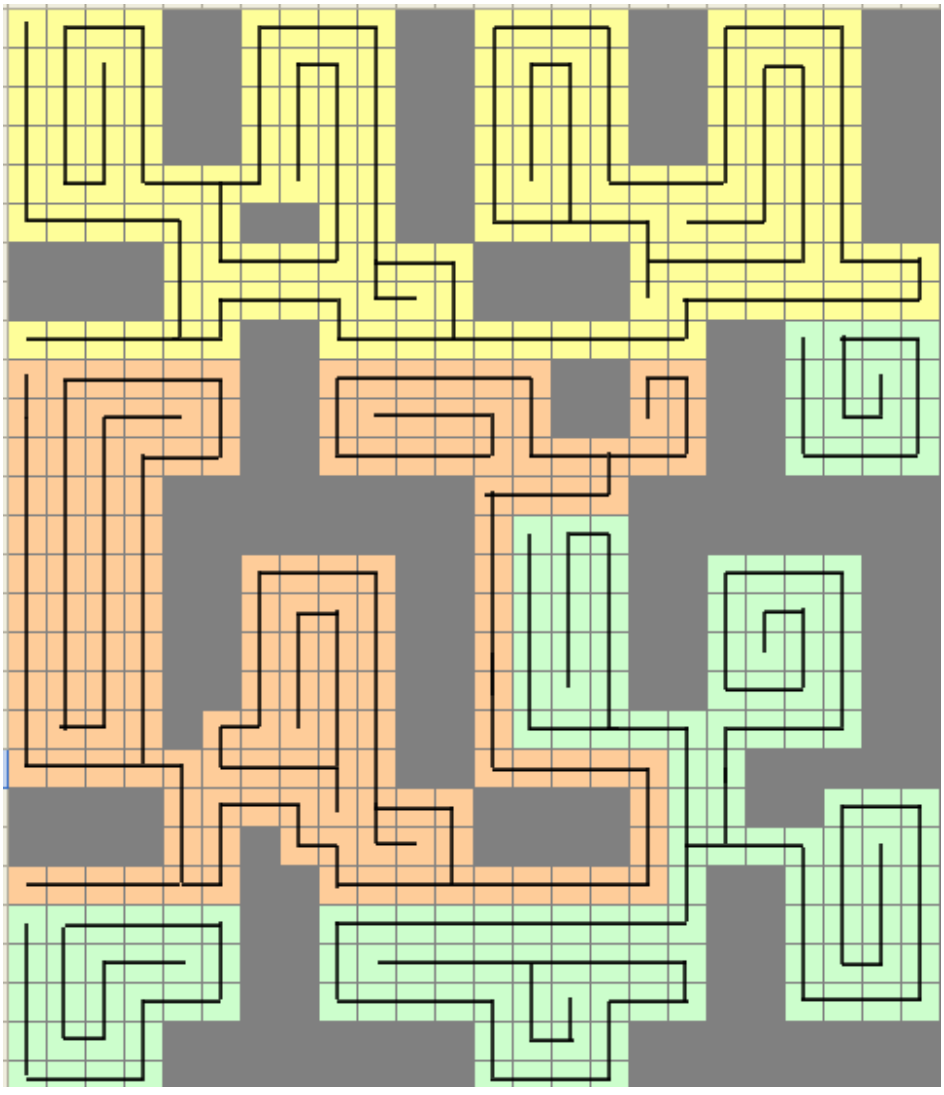
ÇS-KAK algoritması test aşamasında, farklı yerleşim planları üzerinde çalıştırılmıştır. Özellikle büyük alan ve büyük alanı oluşturan dört farklı çalışma alanı üzerinde algoritma test edilmiş, analiz sonucu elde edilen bilgilere izleyen bölümde yer verilmiştir.

4.4.1. Büyük alanda Çoklu Sarmal - Kapsar Ağaç Kapsama algoritması



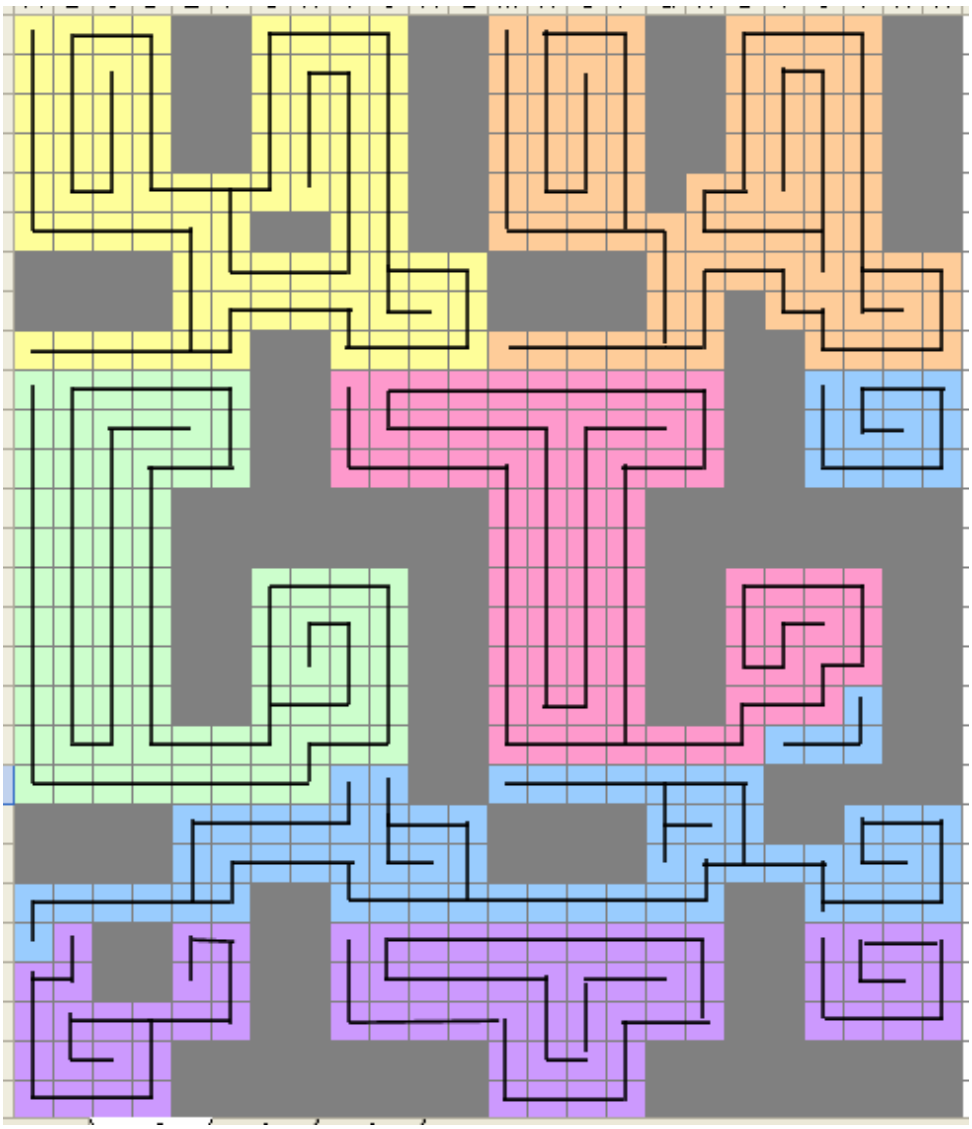
Şekil 4.5 Dört farklı çalışma alanının birleşiminden oluşan büyük alan

Algoritma, üç robot için, dört farklı çalışma alanının birleşiminden oluşan (Şekil 4.5) büyük alanda çalıştırıldığında Şekil 4.6'da yer alan kapsar ağaç yolu elde edilmektedir. Birinci ve ikinci robot kapsar ağaç yollarında tekrarlı kapsama söz konusu olmayıp, üçüncü robot yolunda alan bölünmesi ve bu sebeple oluşan tekrarlı kapsama söz konusudur. Tekrar kapsanan hücre sayısı ise alan bölünmeleri arasındaki en kısa dik doğrusal uzaklık dikkate alındığında 19 hücre olarak karşımıza çıkmaktadır.



Şekil 4.6 Üç robot için büyük alanda ÇS-KAK algoritması

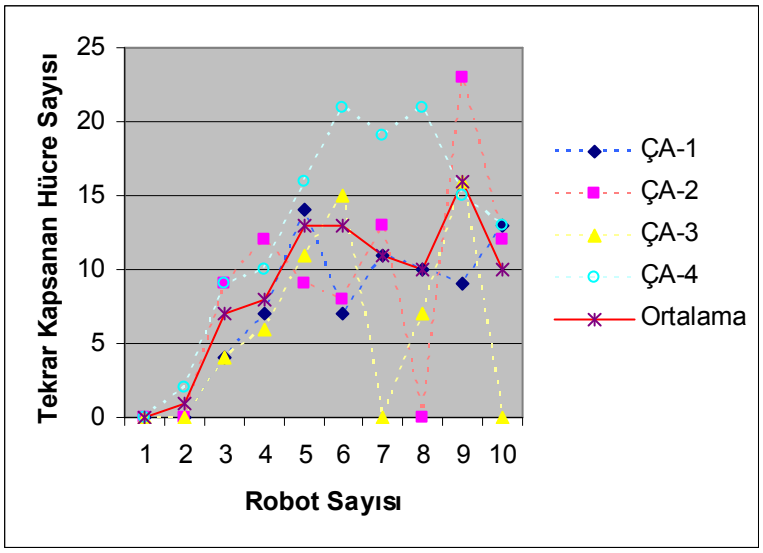
Algoritma, altı robot için, (Şekil 4.5) büyük alanda çalıştırıldığında Şekil 4.7'de yer alan kapsar ağaç yolu elde edilmektedir. Birinci, ikinci, üçüncü ve dördüncü robot kapsar ağaç yollarında tekrarlı kapsama söz konusu olmayıp; beşinci ve altıncı robot yolunda alan bölünmesi ve bu sebeple oluşan tekrarlı kapsama söz konusudur. Tekrar kapsanan hücre sayısı ise alan bölünmeleri arasındaki en kısa dik doğrusal uzaklık dikkate alındığında 33 hücre olarak karşımıza çıkmaktadır.



Şekil 4.7 Altı robot için büyük alanda ÇS-KAK algoritması

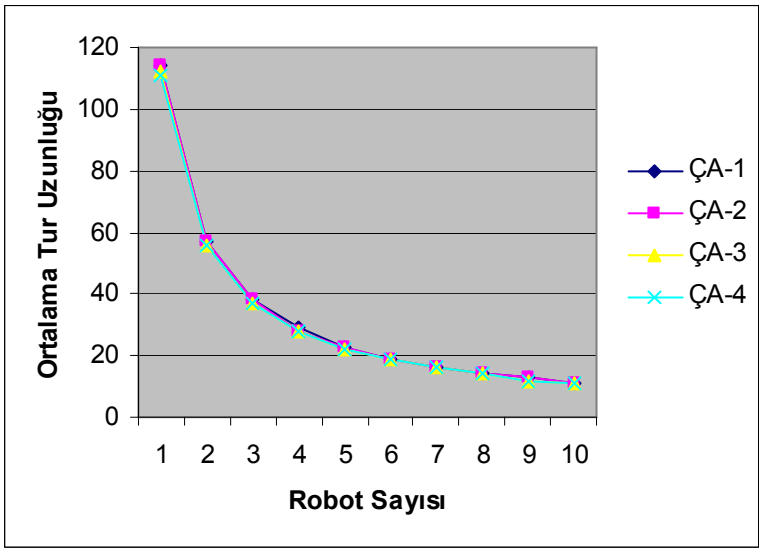
4.4.2. Dört farklı çalışma alanında Çoklu Sarmal - Kapsar Ağaç Kapsama algoritması

Önerilen algoritma,1'den 10'a kadar robot sayıları değiştirilerek dört farklı yerleşimde çalıştırılmış, tekrar kapsanan hücre sayıları ve ortalama tur uzunlukları gözlenmiştir. Tekrar kapsanan hücre sayıları değerlendirilirken robotun kendi alanı üzerindeki herhangi bir hücreden kapsamaya başladığı varsayılmıştır. Şekil 4.7'de dört farklı çalışma alanında elde edilen sonuçlar ve bunların ortalama değerleri grafikle gösterilmiştir.



Şekil 4.8 Dört farklı çalışma alanında, tekrar kapsanan hücre sayısı

Şekil 4.8'de dört farklı çalışma alanı için 1'den 10'a kadar robot sayıları değiştirildiğinde robot başına düşen ortalama tur uzunluğu yer almaktadır. Tur uzunluğu hesabında her hücre 1 birim kabul edilmiştir. Robot sayısı arttıkça tur uzunluğunda azalma oluşurken, 4 robottan sonra yeni robot eklentisinin tur uzunluğundaki etkisi azalmaktadır.



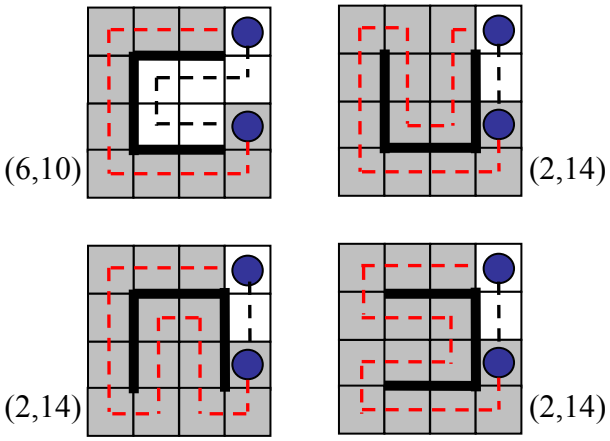
Şekil 4.9 Dört farklı çalışma alanında, ortalama tur uzunluğu

5. ÖNERİLEN YAKLAŞIMIN DİĞER YÖNTEMLERLE KARŞILAŞTIRILMASI

Gabriely ve Rimon'un (2001a) önermiş olduğu, gezgin robotlar için Statik ortamda KAK, Dinamik ortamda KAK ve Karıncamsı KAK algoritmaları ile tek robot için kapsar ağaç yolu oluşturulmuş, çok robot için çözüm sunulmamıştır. (Chang and Dan, 2006) çalışmasında ise yine tek robot için dinamik ortam zigzag KAK ve ilk en büyük dinamik ortam KAK algoritmaları geliştirilmiştir. (Gonzáles, et al., 2005) çalışmasında gezgin robotlar için sarmal yol kullanan kapsama stratejisi olan Geri Dönüşlü Sarmal Algoritmada da tek robot için yol oluşturulmaktadır. Alanın grid hücrelerinin boyutu ile robot boyutu aynı olmakla birlikte tek robot için birden fazla parçalı yol oluşmakta bu yollar arasındaki geçişlerde tekrarlı kapsamalar söz konusu olmaktadır. Adı geçen algoritmalar çalışma alanının yalnız tek robot ile kapsanmasını sağlamakta, tek robot için çözüm oluşturmaktadır; geliştirilen ÇS-KAK algoritması ise hem tek robot, hem de çok robot ile kapsamanın elde edilebildiği kapsar ağaç kapsama yolu oluşturmaktadır. Kapsamayı gerçekleştirecek robot sayısı başlangıçta programa girilerek kullanıcı tarafından belirlenebilmektedir.

5.1. Etkili Çoklu Robot Kapsaması için Kapsar Ağaç Yapılandırması ile Çoklu Sarmal - Kapsar Ağaç Kapsama Algoritmasının Karşılaştırılması

(Agmon, et al., 2006) çalışmasında robotlar arası mesafe eşitlenmeye çalışılarak, alt kapsar ağaç yolları oluşturulmakta ve sonrasında bu alt yollar birleştirme işlemine tabi tutulmaktadır. Ancak yerleşime bağlı olarak robotların kat ettikleri mesafenin eşitlenmesi her zaman mümkün olmayabilmektedir.



Şekil 5.1 Olası yollar içinde [N/k] uzaklığının sağlanamadığı kapsar ağaç yolu

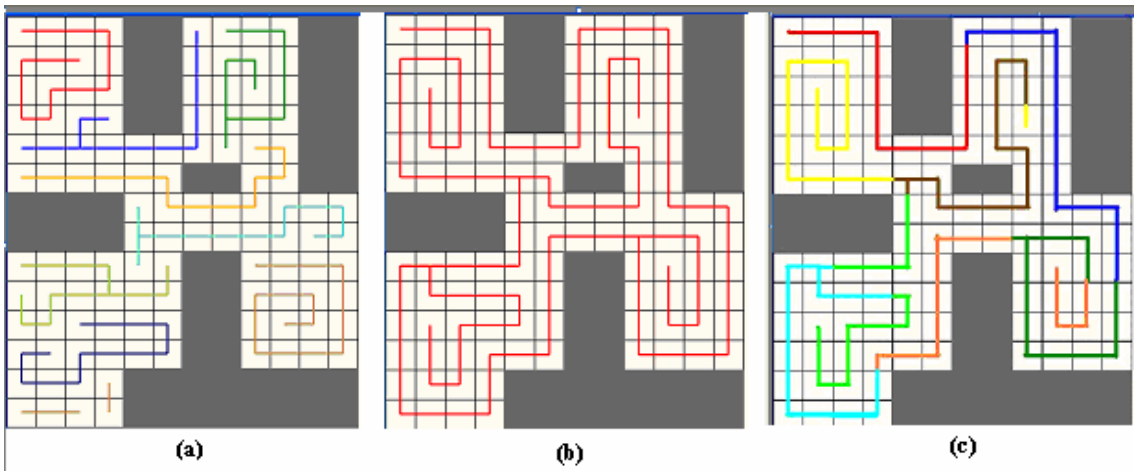
Kapsama süresinin enküçüklenmesi için robotlar arası mesafenin [N/k]'ya eşit olması gerekmektedir. Şekil 5.1'de N=16 hücre ve k=2 robot bulunmaktadır. Robotlar arası mesafe 8 olması gerekirken Şekil5.1'de de görüldüğü üzere bütün olası kapsar ağaç yolları elde edilse bile [N/k] uzaklığı elde edilemeyebilmektedir. ÇS-KAK algoritmasında ise başlangıçta her bir robota eşit hücre atanması suretiyle robotların tarayacakları mesafenin eşitlenmesi sağlanmış durumdadır.

(Agmon, et al., 2006) çalışmasında çoklu robot için oluşturulan kapsar ağaç yolu tektir ve sonuçta tek bir kapsar ağaç yolunun robotlara paylaştırılması söz konusu olmaktadır. ÇS-KAK algoritmasında ise her bir robot için birbirinden bağımsız kapsar ağaç yolları oluşmaktadır. Bu ise izleyen bölümde de ayrıntılandırıldığı gibi (Agmon, et al., 2006) çalışmasına göre büyük bir üstünlük olarak karşımıza çıkmaktadır.

5.2. Çoklu Kapsar Ağaç Kapsama Algoritması ile Çoklu Sarmal - Kapsar Ağaç Kapsama Algoritmasının Karşılaştırılması

Hazon ve Kaminka'nın (2008) önermiş olduğu ÇKAK algoritmasında, robot için başlangıç noktası bir hücre olabiliyorken ve o hücreden başlanmadığında tekrar eden

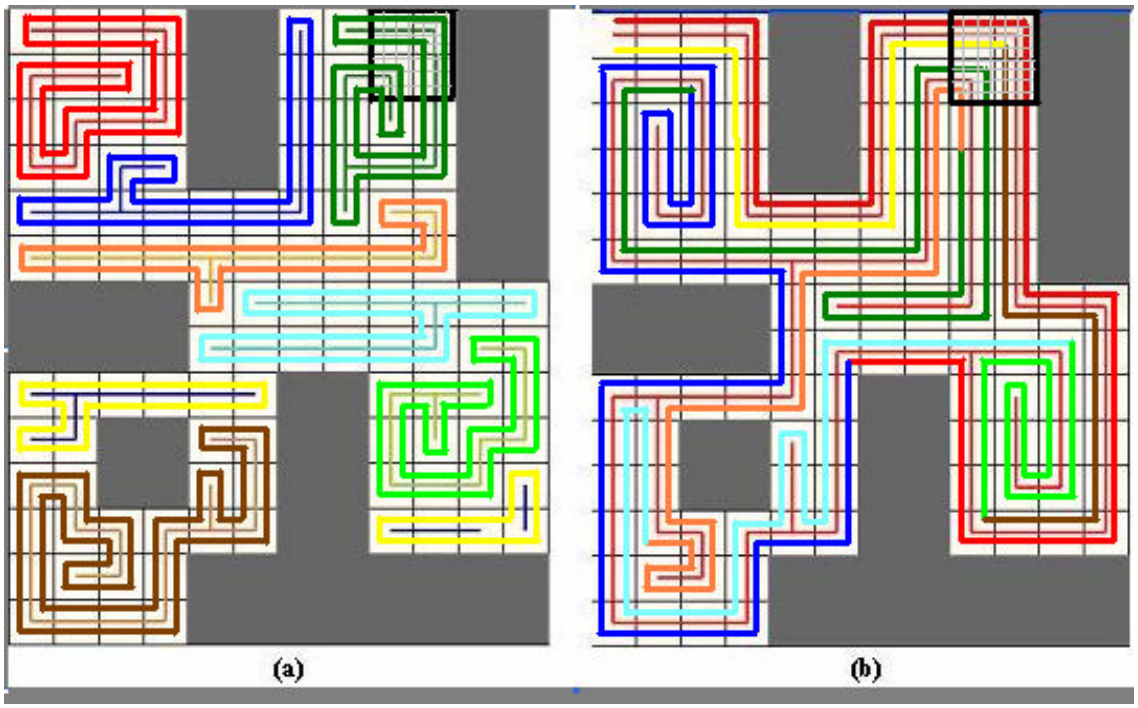
kapsama oluşuyorken, geliştirilen ÇS-KAK algoritmasının başlangıç noktası açısından sunmuş olduğu serbestlik büyük bir avantaj sağlamaktadır.



Şekil 5.2 (a) Sekiz robot için ÇS-KAK algoritmasında kapsar ağaç yolu (b) Tek robot için kapsar ağaç yolu (c) Sekiz robotlu ÇKAK'da eşit hücre paylaşımı

ÇKAK'da bir robot için bulunan kapsar ağaç yolu başlangıç noktaları ataması ile çok robota atanır. Robotlara eşit sayıda hücre atanması kriteri dikkate alınmamış, robotlara eşit iş yükü dağılmamıştır. ÇS-KAK algoritmasında ise robotlara eşit sayıda hücre atandıktan sonra her bir robota ayrı kapsar ağaç yolu oluşturulur. Yeni yöntem ÇKAK'ya uygulanıp, tek bir kapsar ağaç eşit sayıda hücre prensibiyle bölünmek istenirse bile, yol bir çok noktadan bölünmek zorunda kalır ve tek bir robota ait birden fazla kapsar ağaç oluşur. Bu ise tekrar eden kapsamaya ve kapsama süresinde artışa sebep olmaktadır. Şekil5.2'deki (a) seçeneğinde ÇS-KAK algoritmasında sekiz robot için eşit sayıda hücre prensibiyle bölünen alanın kapsar ağaç yolları gözlenmektedir. Sadece son robotta alan bölünmesi söz konusudur. ÇKAK'da ise (b) seçeneğinde yer alan tek robot için oluşturulan kapsar ağaç yolu (c) seçeneğinde yer alan şekilde de görüldüğü gibi sekiz robota paylaştırılmaktadır. ÇKAK yaklaşımına eşit hücre paylaşımı uygulandığında 4., 6. ve 8. robot için alan bölünmesi oluşmaktadır. Bu da çok fazla robot için verimsiz kapsar ağaç yolu demektir. Bu örnekte de görüldüğü gibi,

önerilen algoritmanın bir diğer üstünlüğü çok robot için eşit iş yükü oluşturması ve kapsamanın tamamlanma zamanını enküçükleyebilmesidir.

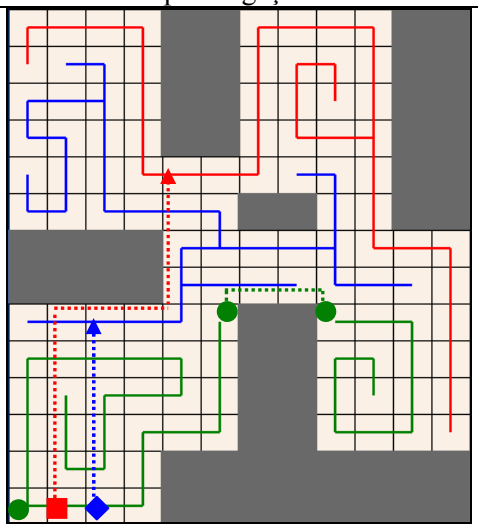
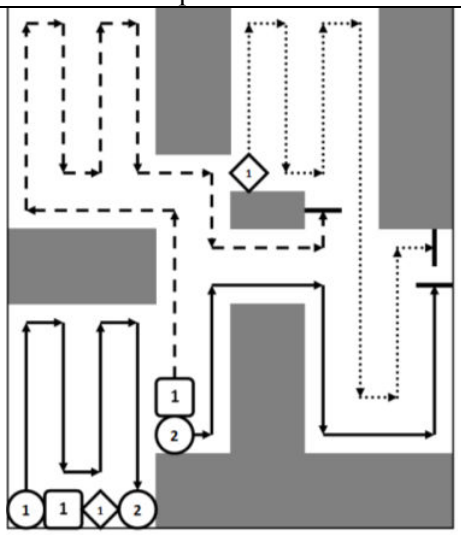


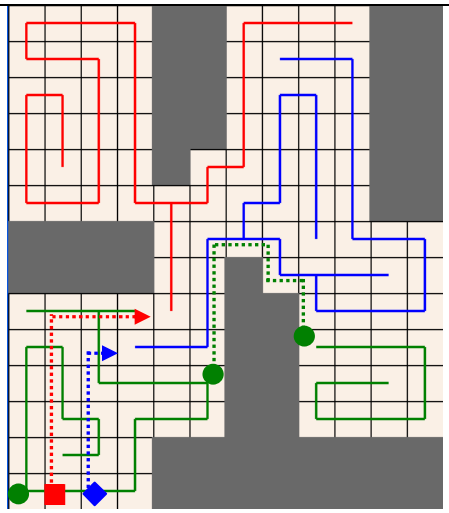
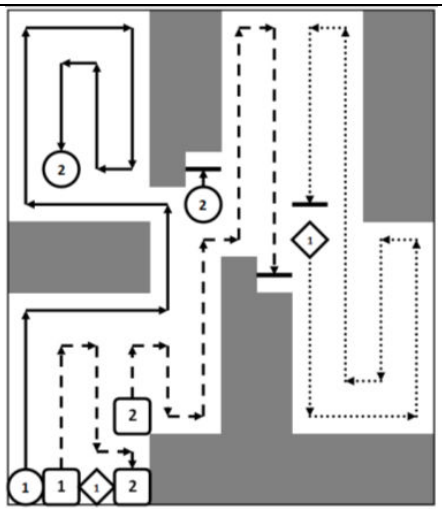
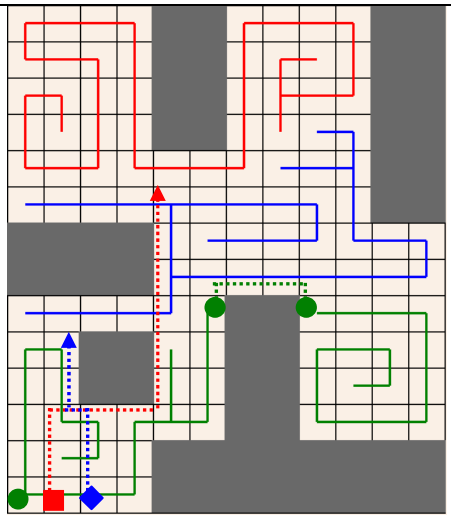
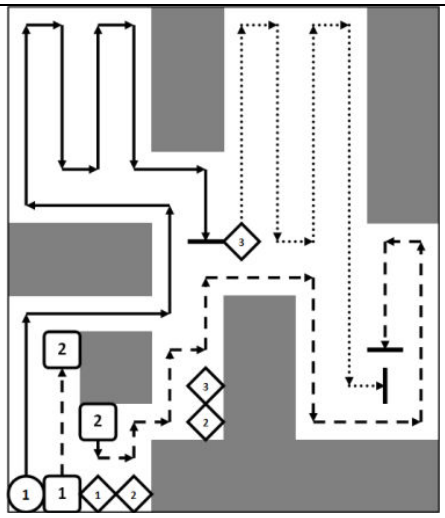
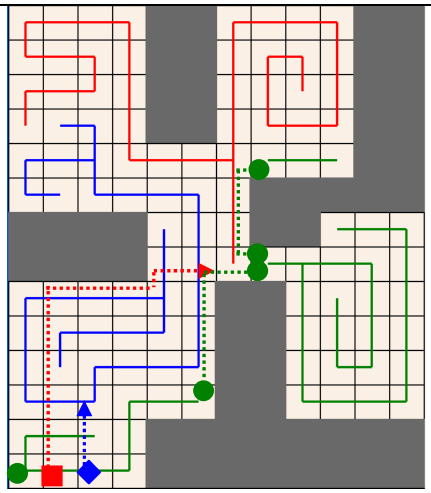
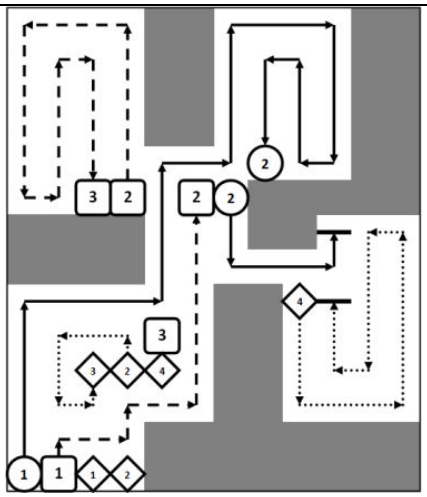
Şekil 5.3 (a) Sekiz robot için kapsar ağaç yolu (b) ÇKAK ile sekiz robot için kapsar ağaç yolu

Daha önce boş olan bir hücreye engel yerleştirildiğinde ÇKAK'da oluşturulan robot yollarında birden fazla robotun yolu engellenebilmektedir. Şekil 5.3'de de örneği görüldüğü üzere, yeni yöntemle, Şekil5.3(a)'daki sağ üst köşedeki dört hücreye engel yerleştirildiğinde, sadece yeşil ile gösterilen robot yolu engellenmektedir. Şekil5.3(b)'deki ÇKAK'da ise kırmızı, yeşil, kahverengi, turuncu, sarı ile gösterilen beş robotun yolu engellenmektedir. Bu ise önerilen algoritma ile daha güvenli bir kapsama elde edildiğinin göstergesi olmaktadır. ÇS-KAK algoritması ile çalışma alanında olabilecek değişikliklere uyum sağlama oranı da artmaktadır.

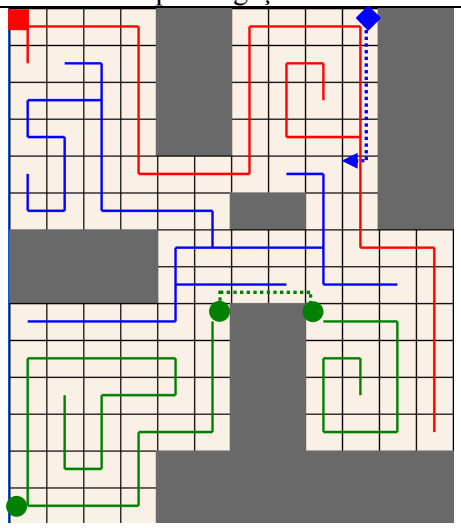
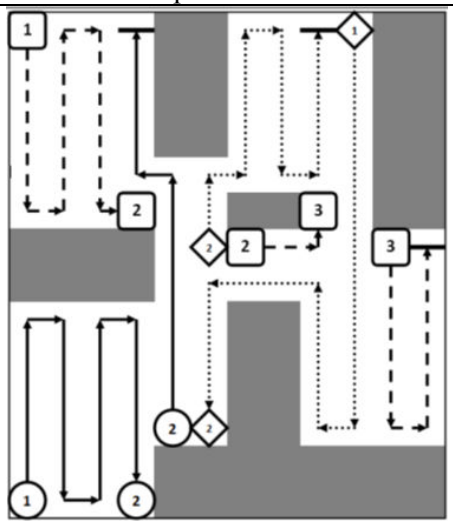
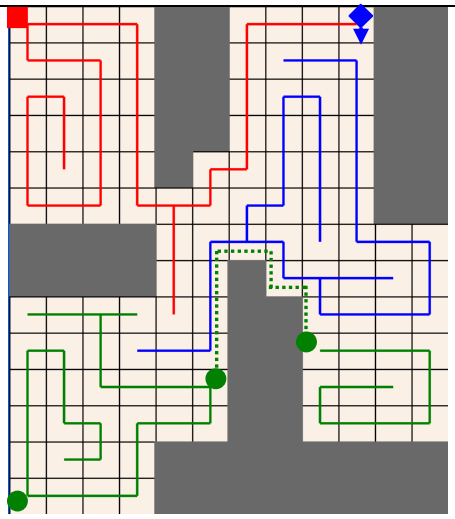
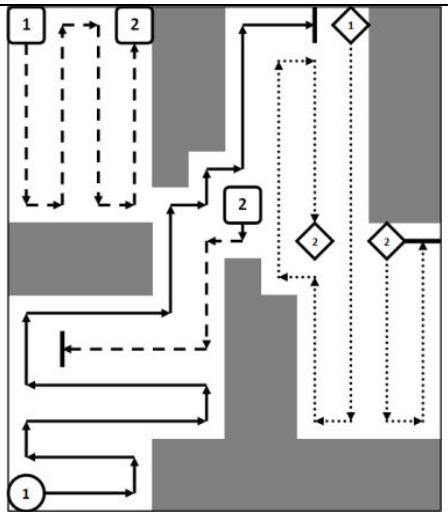
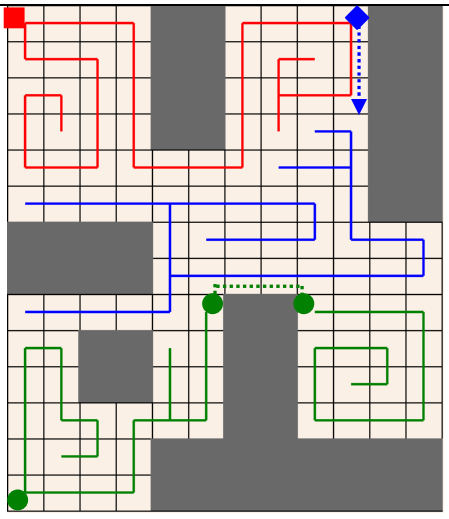
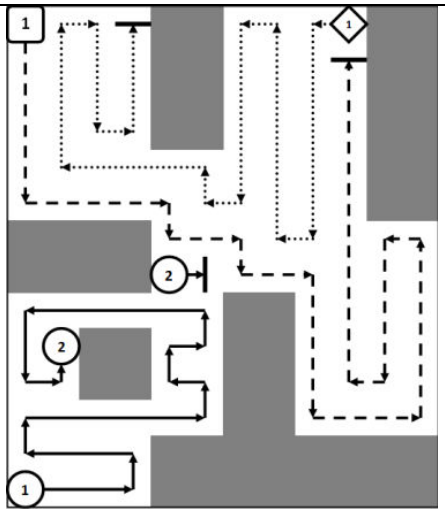
5.3. Çoklu Robot Kapsamasında Genetik Algoritma ile Çoklu Sarmal - Kapsar Ağaç Kapsama Algoritmasının Karşılaştırılması

Örüntü tabanlı genetik algoritmada (Kapanoğlu vd., 2010), robotların başlangıç noktası belirlendikten sonra, yerleşime en uygun kapsama yolu oluşturulmaktadır. Önerilen ÇS-KAK algoritmasında ise başlangıç noktasından bağımsız olarak kapsama yolları oluşturulur. Her robot için ayrı olarak oluşturulan kapsar ağaç yolu, kapalı eğri şeklinde olan yollar türetir. Bu sayede robotlar taramaya başladıkları noktaya geri dönmektedir. Elde edilen çözümde robota ait her hücre,tekrarlı kapsama olmaksızın (robot yolu bölünmediği sürece) o robot için alternatif başlangıç noktası olmaktadır. Böylelikle, algoritmada her ayrı başlangıç noktası için çözüm aranmak zorunda kalmamakta, robot sayısına bağlı olarak elde edilen çözüm, başlangıç noktası açısından sunduğu serbestlik sayesinde, farklı başlangıç pozisyonlarına uyum sağlayabilmektedir. Şekil 5.4'de robotlar aynı yerden taramaya başladıklarında, iki algoritma için, dört farklı çalışma alanında oluşan kapsar ağaç yolları gözlenmektedir.

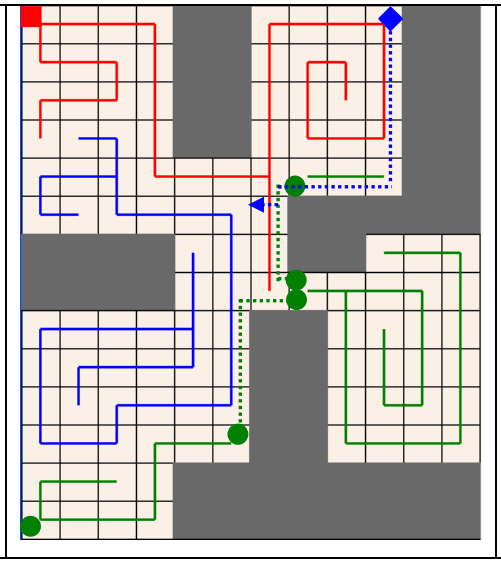
	Robotların Başlangıç Noktası Depo	
	ÇS-KAK Algoritması İçin Kapsar Ağaç Yolu	Genetik Algoritma ile Kapsama Yolu
ÇA-1		

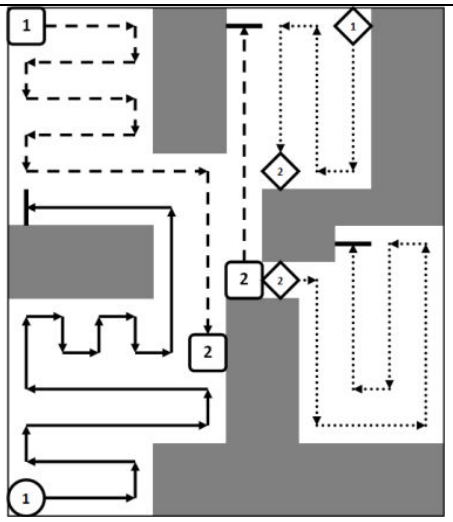
ÇA-2		
ÇA-3		
ÇA-4		

Şekil 5.4 Robotların başlangıç pozisyonu depo iken ÇS-KAK algoritması ve genetik algoritma için kapsama yolu

	Robotların Başlangıç Noktası Rastgele Belirlenmiş	
	ÇS-KAK Algoritması İçin Kapsar Ağaç Yolu	Genetik Algoritma ile Kapsama Yolu
ÇA-1		
ÇA-2		
ÇA-3		

ÇA-4





Şekil 5.5 Robotlar rastgele belirlenmiş başlangıç noktasında iken ÇS-KAK algoritması ve genetik algoritma için elde edilen çözüm

Şekil 5.5'de robotlar rastgele belirlenmiş başlangıç noktalarında iken, iki algoritma için, dört farklı çalışma alanında elde edilen çözümler yer almaktadır.

Çizelge 5.1 Dört farklı çalışma alanında, farklı başlangıç noktalarında ÇS-KAK algortiması ve genetik algoritma ile elde edilen çözümlerde tekrar kapsanan hücre sayıları

	Robotların Başlangıç Noktası Depo İken		Robotların Başlangıç Noktası Rastgele Belirlenmiş İken	
	ÇS-KAK Algoritması Tekrar Kapsanan Hücre Sayısı	Genetik Algoritma Tekrar Kapsanan Hücre Sayısı	ÇS-KAK Algoritması Tekrar Kapsanan Hücre Sayısı	Genetik Algoritma Tekrar Kapsanan Hücre Sayısı
ÇA-1	20	22	6	15
ÇA-2	22	20	10	11
ÇA-3	20	12	7	5
ÇA-4	21	23	18	8

Çizelge 5.1'de dört farklı yerleşimde, ÇS-KAK algoritması ve genetik algoritma ile elde edilen çözümlerde, tekrar kapsanan hücre sayısı toplamı verilmiştir. Çizelgede de görüldüğü üzere, her iki algoritmada elde edilen çözümde tekrar kapsanan hücre sayıları birbirine yakın değerlerde elde edilmektedir. Ek olarak genetik algoritmada her bir farklı başlangıç pozisyonu için ayrı çözüm oluşturma gerekliliği ve çözüm süresinin uzun olması örüntü tabanlı genetik algoritmanın dezavantajı olarak karşımıza çıkmaktadır.

6. SONUÇ VE ÖNERİLER

Araştırma kapsamında, çoklu gezgin robotların statik çalışma alanını tam kapsaması yol planı oluşturan yeni bir kapsar ağaç kapsama (ÇS-KAK) algoritması geliştirilmiştir. ÇS-KAK algoritması alan içindeki gezilmesi gerekli bölümleri robotlara eşit olarak paylaştırmakta böylece robotlara eşit iş yükü dağılmakta, böylelikle kapsama süresinde de iyileşme sağlanmaktadır. Ayrıca robotun kapsama işlemine kendisine atanan herhangi bir hücreden başlayabiliyor olması ve bu sayede çok sayıda olası başlangıç noktası seçeneği oluşturması algoritmanın sunduğu büyük bir avantaj olarak elde edilmektedir. Çalışma alanındaki boş bir hücreye engel yerleştirildiğinde önerilen yöntemle sadece bir robotun yolunun engelleniyor olması, algoritmanın ortamdaki değişikliklere karşı uyum sağlayabilme yeteneğinin fazla olduğunun ve dolayısı ile daha güvenli bir kapsama elde edildiğinin göstergesi olmaktadır. Aynı zamanda geliştirilen algoritma Visual Studio 2008 ortamında C# dili kullanılarak kodlanmış olup, çözüm süresi göz ardı edilebilecek kadar kısadır. Bu yaklaşım genetik algoritma gibi çözüm elde etmenin uzun süreler alabildiği, alanda gerçekleştirilen diğer çalışmalara göre büyük bir avantaj sağlamaktadır.

Enküçük sarmal kapsar ağaç yol planlama algoritması; robotlar için eşit sayıda dönüş ölçütünü içeren, robotlar için şarj noktaları kısıtı söz konusu olduğunda ideal başlangıç noktalarını bulan, çalışma alanı boyutuna göre sarmal yapıya uygun ideal robot sayısını bulan, robotların birbirlerinin yerini alması gerektiği durumlarda yerine geçme prosedürü için yol gösteren, statik çalışma alanı için önerilen yöntemin genişletildiği algoritmalar gibi pek çok yönde geliştirilebilir durumdadır. Statik çalışma alanı için çok robot kapsar ağaç tam kapsamasında daha az tekrarlı kapsama oluşturan bu araştırmanın üzerinde çalışılabilecek pek çok konu üretmesi yönüyle de gelecek çalışmalara ve robotik alanında çalışanlara ilham kaynağı olacağı düşünülmektedir.

KAYNAKLAR

Agmon, N., Hazon, N. and Kaminka, G.A., 2006, Constructing Spanning Trees for Efficient Multi-Robot Coverage, Proceedings of the 2006 IEEE International Conference on Robotics and Automation, 1698-1703.

Bosse, M., Vatani, N.N. and Roberts, J., 2007, Coverage Algorithms for an Under-actuated Car-Like Vehicle in an Uncertain Environment, 2007 IEEE International Conference on Robotics and Automation, 698-703.

Chaing, S.J. and Dan, B.J., 2006, Free moving pattern's Online Spanning Tree Coverage Algorithm, SICE-ICASE International Joint Conference, 2935-2938.

Choset, H., 2000, Coverage of Known Spaces: The Boustrophedon Cellular Decomposition, Autonomous Robots 9, 247–253.

Choset, H., 2001, Coverage for robotics – A survey of recent results, Annals of Mathematics and Artificial Intelligence 31, 113–126.

Choset, H., Lynch, K.M., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L.E. and Thrun, S.,2005, Principles of Robot Motion, A Bradford Book The MIT Press, London, 603 p.

Gabriely, Y. and Rimon, E., 2001, Spanning-tree based coverage of continuous areas by a mobile robot, Annals of Mathematics and Artificial Intelligence 31, 77–98 (a).

Gabriely, Y. and Rimon, E., 2001, Spiral-STC: An On-Line Coverage Algorithm of Grid Environments by a Mobile Robot, Proceedings of the 2002 IEEE International Conference on Robotics, 954-960 (b).

Garcia, E. and Gonzalez de Santos, P., 2004, Mobile-robot navigation with complete coverage of unstructured environments, Robotics and Autonomous Systems 46, 195–204.

González, E., Álvarez, O., Díaz, Y., Parra, C. and Bustacara, C., 2005, BSA: A Complete Coverage Algorithm, Proceedings of the 2005 IEEE International Conference on Robotics and Automation, 2040-2044.

Hazon, N. and Kaminka, G.A., 2005, Redundancy, Efficiency and Robustness in Multi-Robot Coverage, Proceedings of the 2005 IEEE International Conference on Robotics and Automation, 735-741.

Hazon, N.and Kaminka, G.A., 2008, On redundancy, efficiency, and robustness in coverage for multiple robots, Robotics and Autonomous Systems 56, 1102–1114.

Hazon, N., Mieli, F. and Kaminka, G.A., 2006, Towards Robust On-line Multi-Robot Coverage, Proceedings of the 2006 IEEE International Conference on Robotics and Automation, 1710–1715.

Huang, R. and Z´aruba, G.V., 2007, Static Path Planning for Mobile Beacons to Localize Sensor Networks, Proceedings of the Fifth Annual IEEE International Conference on Pervasive Computing and Communications Workshops, 6 p.

Huang, W.H., 2001, Optimal Line-sweep-based Decompositions for Coverage Algorithms, Proceedings of the 2001 IEEE International Conference on Robotics &Automation, 27-32.

Kapanoğlu, M., Alikalfa, M., Özkan, M., Yazıcı, A. and Parlaktuna, O., , A Pattern-Based Genetic Algorithm For Multi-Robot Coverage Path Planning Minimizing Completion Time, Journal of Intelligent Manufacturing (Basılmak Üzere Kabul Edildiği Tarih: 4 Nisan 2010).

Kapanoğlu, M., Özkan, M., Yazıcı, A. and Parlaktuna, O., 2009, Pattern-Based Genetic Algorithm Approach to Coverage Path Planning for Mobile Robots, ICCS 2009, 33-42.

Kim, S.J., Kang J.W. and Myung Jin Chung, 2007, Efficient Area Coverage Method for a Mobile Robot in Indoor Environments, The 4th International Conference on Ubiquitous Robots and Ambient Intelligence, 580-585.

Kirlik, G., 2009, Yeni bir kapasiteli ayrıt rotalama problemi ve çözüm yaklaşımları, Yüksek lisans tezi, O.G.Ü. Endüstri Mühendisliği Anabilim Dalı, 136 s.

Kong, C.S., Peng, N.A. and Rekleitis, J., 2006, Distributed Coverage with Multi-Robot System, Proceedings of the 2006 IEEE International Conference on Robotics and Automation, 2423-2429.

Liu, Y. and Zhu, X.L.S., 2008, Combined Coverage Path Planning for Autonomous Cleaning Robots in Unstructured Environments, Proceedings of the 7th World Congress on Intelligent Control and Automation, 8271-8276.

Mei, Y., Lu, Y.H., Lee, C.S.G., and Hu, Y.C., 2006, Energy-Efficient Mobile Robot Exploration, Proceedings of the 2006 IEEE International Conference on Robotics and Automation, 505-511.

Oh', J.S., Choi., Y.H., Park', J.B. and Zheng, Y.F., 2003, Navigation of Cleaning Robots Using Triangular-Cell Map for Complete Coverage, Proceedings of the 2003 IEEE International Conference on Robotics &Automation, 2006-2011.

Özkan, M., Yazıcı, A., Kapanoğlu, M. and Parlaktuna, O., 2009, A genetic algorithm for task completion time minimization for multi-robot sensor-based coverage, Part of 2009 IEEE Multi-conference on Systems and Control, 1164-1169.

Parlaktuna, O., Sipahioğlu, A., Kirlik, G. and Yazıcı, A., 2009, Multi-robot sensor-based coverage path planning using capacitated arc routing approach, Part of 2009 IEEE Multi-conference on Systems and Control, 1146-1151.

Surve, S., Singh, N.M. and Lande, B.K., 2007, CPPA: A Fast Coverage Algorithm, International Conference on Computational Intelligence and Multimedia Applications, 151-156.

Yao, Z., 2006, Finding Efficient Robot Path for the Complete Coverage of A Known Space, Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, 3369-3374.

Yazıcı, A., Kirlik, G., Parlaktuna, O. and Sipahioğlu, A., 2009, A Dynamic Path Planning Approach for Multi-Robot Sensor-Based Coverage Considering Energy Constraints, The 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, 5930-5935.

Yufka, A., Yazıcı, A. and Parlaktuna, O., 2009, A Smooth Path Generation Approach for Sensor-based Coverage Path Planning, 375-379.

EKLER

Ek. 1. Kruskal ve Prim Algoritması

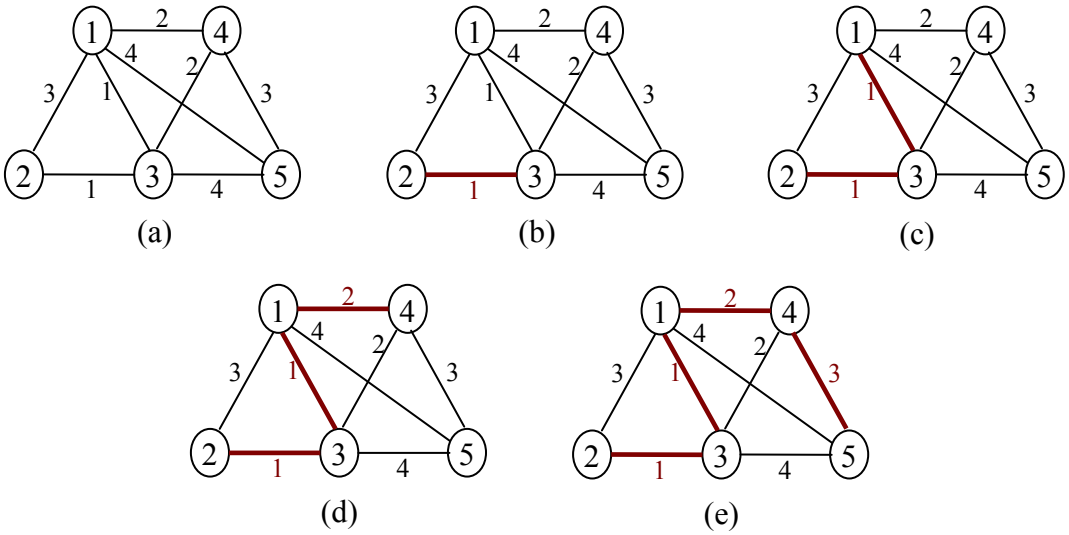
Ek. 2. ÇS-KAK Algoritması C# Kodları

Ek. 1. Kruskal ve Prim Algoritması

Kruskal Algoritması

Kruskal algoritması, bir serim üzerinde en küçük kapsar ağaç bulmak için kullanılan algoritmalardan birisidir. Algoritma adımları;

1. Başlangıçta ağaç hiç ayrıt içermez.
2. Daha önceden ağaca katılmamış en küçük ağırlıklı ayrıt seçilir.
3. Eğer bu ayrıtın ağaca katılması, bir halka oluşmasına sebep olmuyorsa ağaca katılır. (Seçilen bir ayrıtın her iki ucundaki düğümler, daha önce ağaca eklenmiş düğümlerin uçlarıysa, bu ayrıtı ağaca eklediğimizde bir halka oluşacaktır. Bu yüzden böyle bir ayrıt ağaca eklenmemelidir)
4. Ağaçtaki ayrıt sayısı (N-1)'e ulaşana kadar ikinci adıma geri dönülür.



Prim Algoritması

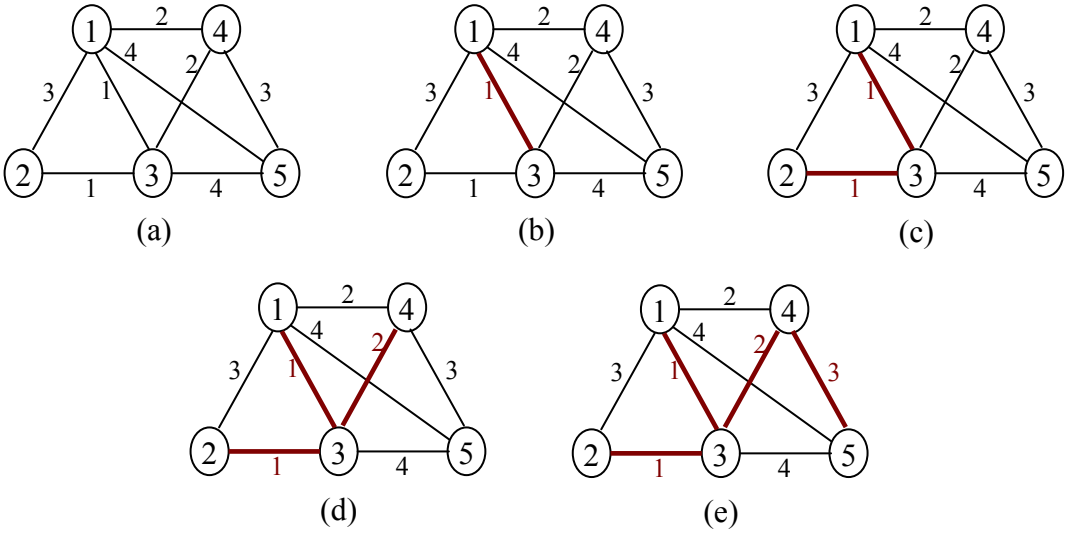
Prim algoritması, bir serim üzerinde en küçük kapsar ağaç bulmak için kullanılan algoritmalardan diğeridir. Algoritma adımları;

1. Başlangıçta herhangi düğümü ağacı oluşturmaya başlamak için seç. (başlangıç düğümü olarak en küçük ağırlıklı ayrıt ucundaki düğüm seçilerek algoritma daha verimli hale getirilebilir.)

2. Oluşturulan ağaca eklemek için, şu ana kadar oluşturulmuş ağaç üzerinden erişilebilen ve daha önceden ağaca katılmamış olan en küçük ağırlıklı ayrıtı seç.

3. Eğer bu ayrıtın ağaca katılması, bir halka oluşmasına sebep olmuyorsa, ağaca ekle. (Seçilen bir ayrıtın her iki ucundaki düğümler, daha önce ağaca eklenmiş düğümlerin uçlarıysa, bu ayrıtı ağaca eklediğimizde bir halka oluşacaktır. Bu yüzden böyle bir ayrıt ağaca eklenmemelidir)

4. Ağaçtaki ayrıt sayısı (N-1)'e ulaşana kadar (tüm düğümler ağaca dahil edilene kadar) ikinci adıma geri dön.



Prim algoritmasının icra zamanı sadece düğümlerin sayısına bağlıdır, fakat aynı sayıda düğümlü bir serim için ayrıtların sayısı artarken Kruskal'ın algoritmasının icra

zamanı artar. Bununla birlikte genelde hangisinin daha verimli olduğunu söylemek mümkün değildir. Verimlilik, ağın yapısına ve ağırlığın dağılımına bağlıdır. Çoğu durumda oluşturulan veri yapısı verimliliğe doğrudan etkilidir.

Ek. 2. ÇS-KAK Algoritması C# Kodları

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Collections;
using System.IO;
using Excel;
using Microsoft.Office.Core;

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        private Excel.Application ExcelNesnesi = null;

        public Form1()
        {
            InitializeComponent();
            ExcelNesnesi = new Excel.Application();
            if (ExcelNesnesi == null)
            {
                MessageBox.Show("Problem! Dosya Açılamadı.");
                System.Windows.Forms.Application.Exit();
            }
        }

        private void Form1_Load(object sender, EventArgs e)
        {

        }

        private void btnCikis_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        #region SINIFLAR
        public class Degisken
        {
            public int HucreSayisi = 12 * 14;
            public int[,] y = new int[14, 12];
            public int[, , ,] z = new int[14, 12, 14, 12];
            public int[,] t = new int[14, 12];
            public int[,] v = new int[14, 12];
        }
        #endregion

        private void btnHesapla_Click(object sender, EventArgs e)
        {
```

```
            int[,] dugum = new int[14, 12];
            int[,] yedekdugum = new int[14, 12];
            int engelsay=0, yol = 0;
            decimal say = 0, sayac = 0;
            int maxX=0 , maxY=0, maxt=0, max=0;

            #region Gridden veya Excel'den veri girme
            if (sGrdDugum.Visible == true)
            {
                for (int sGrdi = 0; sGrdi < 14; sGrdi++)
                    for (int sGrdj = 0; sGrdj < 12; sGrdj++)
                    {
                        if (sGrdDugum[sGrdi, sGrdj].ToString().Trim()
== "0")
                        {
                            dugum[sGrdi, sGrdj] = 0;
                            engelsay = engelsay + 1;
                        }
                        else
                            dugum[sGrdi, sGrdj] = 100;
                    }
            }
            else if(dataGridView1.Visible == true)
            {
                for (int stn = 0; stn < 12; stn++)
                    for (int str = 0; str < 14; str++)
                    {
                        if (dataGridView1[stn,
str].Value.ToString().Trim() == "0")
                        {
                            dugum[str, stn] = 0;
                            engelsay = engelsay + 1;
                        }
                        else
                            dugum[str, stn] = 100;
                    }
            }
            else
                for (int si = 0; si < 14; si++)
                    for (int sj = 0; sj < 12; sj++)
                        dugum[si, sj] = 100;
            for (int si = 0; si < 14; si++)
                for (int sj = 0; sj < 12; sj++)
                    yedekdugum[si, sj] = dugum[si, sj];
            #endregion

            #region Alanı robotlara ayırma
            decimal robotsay=1;
            int engel=100, atama = 0, sonrobot = 0, ENIYISATIR = 0,
ENIYISUTUN = 0, ENIYITEKRAR = 10000, TEKRAR=0, SONSTN=0,SONSTR=0;
            if (txtRobotSayisi.Text.Trim() != string.Empty &&
Convert.ToInt32(txtRobotSayisi.Text)<=10)
                robotsay = Convert.ToDecimal(txtRobotSayisi.Text);
            else
                MessageBox.Show("Robot sayısı en fazla 10 olacak
şekilde robot sayısını giriniz!", "Robot Sayısında Hata",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
```

```
            say = Convert.ToDecimal(12 * 14 - engelsay) /
(decimal)robotsay; (decimal)robotsay;
            sayac += 1;
            sonrobot = 1;

            for (int DONGUt = 0; DONGUt < 14; DONGUt++)
            {
                for (int DONGUs = 0; DONGUs < 12; DONGUs++)
                {
                    if (dugum[DONGUt, DONGUs] != 0)
                    {
                        sonrobot = 1;
                        sayac = 1;
                        TEKRAR = 0;
                        for (int si = 0; si < 14; si++)
                            for (int sj = 0; sj < 12; sj++)
                                dugum[si, sj] = yedekdugum[si, sj];
                        dugum[DONGUt, DONGUs] = 1;
                        for (int t = 0; t < 14 * 12; t++)
                        {
                            for (int s = 0; s < 14 * 12; s++)
                            {
                                atama = 0;
                                for (int Grdi = 0; Grdi < 14; Grdi++)
                                    for (int Grdj = 0; Grdj < 12;
Grdj++)
                                    {
                                        if (dugum[Grdi, Grdj] == 100
&& atama == 0)
                                        {
                                            for (int rbt = 1; rbt <=
robotsay; rbt++)
                                            {
                                                if
(Convert.ToDecimal(say * (decimal)(rbt - 1)) < (decimal)sayac &&
(decimal)sayac < Convert.ToDecimal(say * (decimal)rbt + (decimal)0.5))
                                                {
if ((rbt != 1 && Convert.ToDecimal(say * (decimal)(rbt - 2)) <
(decimal)(sayac - 1) && (decimal)(sayac - 1) <= Convert.ToDecimal(say
* (decimal)(rbt-1) + (decimal)0.5)) || (Grdi != 0 && dugum[Grdi - 1,
Grdj] == rbt) || (Grdj != 0 && dugum[Grdi, Grdj - 1] == rbt) || (Grdi
!= 13 && dugum[Grdi + 1, Grdj] == rbt) || (Grdj != 11 && dugum[Grdi,
Grdj + 1] == rbt))
                                                    {
                                                        dugum[Grdi,
Grdj] = rbt;

                                                        SONSTR = Grdi;
                                                        SONSTN = Grdj;
                                                        sonrobot =rbt;
                                                        sayac += 1;
                                                        atama = 1;
                                                    }
                                                }
                                            }
                                            if (sayac >(say*robotsay))
                                            {
                                                dugum[Grdi, Grdj] =
Convert.ToInt32(robotsay);
```

```
                                    SONSTR = Grdi;
                                    SONSTN = Grdj;
                                }
                            }
                        }
                    }
                    atama = 0;

                    if (sayac != ((14 * 12) - engelsay))
                    {
                        engel = 100;
                        for (int Grdi = 0; Grdi < 14; Grdi++)
                            for (int Grdj = 0; Grdj < 12;
Grdj++)

                            {
                                if (dugum[Grdi, Grdj] == 100
&& atama == 0)

                                {
                                    dugum[Grdi,Grdj]=sonrobot;
                                    if (SONSTR > Grdi)
                                        TEKRAR += (SONSTR -
Grdi) * 2;

                                    else
                                        TEKRAR += (Grdi -
SONSTR) * 2;

                                    if (SONSTN > Grdj)
                                        TEKRAR += (SONSTN -
Grdj) * 2;

                                    else
                                        TEKRAR += (Grdj -
SONSTN) * 2;


                                    if (SONSTR > Grdi &&
SONSTN > Grdj)

                                    {
                                        int yinesay=0,
boshucresay=0, gec=0;

                                        for (int yinei = Grdi;
yinei <= SONSTR; yinei++)

                                        {
                                            yinesay=SONSTN-
Grdj;

                                            boshucresay=0;
                                            for (int yinej =
Grdj; yinej <= SONSTN; yinej++)

                                            {
                                                if
(dugum[yinei, yinej] != 0)

                                                {
boshucresay += 1;


                                                }
                                            }
                                            if (yinesay ==
boshucresay)

                                                gec = 1;
                                        }
```

```
!= 0)

2;

Grdj; yinej <= SONSTN; yinej++)

1, yinej] == 0 && dugum[ii, yinej] == 0)

> SONSTN)

boshucresay = 0, gec = 0;

yinei <= SONSTR; yinei++)

SONSTN;

SONSTN ; yinej <= Grdj; yinej++)

(dugum[yinei, yinej] != 0)

boshucresay += 1;

boshucresay)

!= 0)

2;

SONSTN; yinej <= Grdj; yinej++)
```

```
    boshucresay = 0;
    int ii = Grdi;
    while (gec == 0 && ii

    {
        TEKRAR = TEKRAR +

        ii -= 1;
        for (int yinej =

        {
            gec = 1;
            if (dugum[ii +

            {
                gec = 0;
            }
        }
    }
}

if (SONSTR > Grdi && Grdj

{
    int yinesay = 0,

    for (int yinei = Grdi;

    {
        yinesay = Grdj -

        boshucresay = 0;
        for (int yinej =

        {
            if

            {


            }
        }
        if (yinesay ==

            gec = 1;
    }
    boshucresay = 0;
    int ii = Grdi;
    while (gec == 0 && ii

    {
        TEKRAR = TEKRAR +

        ii -= 1;
        for (int yinej =

        {
            gec = 1;
```

```
1, yinej] == 0 && dugum[ii, yinej] == 0)
```

```
SONSTN  > Grdj)
```

```
boshucresay = 0, gec = 0;
```

```
SONSTR; yinei <= Grdi; yinei++)
```

```
Grdj;
```

```
Grdj; yinej <= SONSTN; yinej++)
```

```
(dugum[yinei, yinej] != 0)
```

```
boshucresay += 1;
```

```
boshucresay)
```

```
!= 13)
```

```
2;
```

```
Grdj; yinej <= SONSTN; yinej++)
```

```
1, yinej] == 0 && dugum[ii, yinej] == 0)
```

```
> SONSTN)
```

```
                if (dugum[ii +
                {
                    gec = 0;
                }
            }
        }
    }

    if (Grdi > SONSTR &&
    {
        int yinesay = 0,
        for (int yinei =
        {
            yinesay = SONSTN -
            boshucresay = 0;
            for (int yinej =
            {
                if
                {

                }
            }
            if (yinesay ==
                gec = 1;
        }
        boshucresay = 0;
        int ii = Grdi;
        while (gec == 0 && ii
        {
            TEKRAR = TEKRAR +
            ii += 1;
            for (int yinej =
            {
                gec = 1;
                if (dugum[ii -
                {
                    gec = 0;
                }
            }
        }
    }

    if (Grdi > SONSTR && Grdj
    {
```

```
                                                    int yinesay = 0,
boshucresay = 0, gec = 0;
                                                    for (int yinei =
SONSTR; yinei <= Grdi; yinei++)
                                                    {
                                                        yinesay = Grdj -
SONSTN;
                                                        boshucresay = 0;
                                                        for (int yinej =
SONSTN; yinej <= Grdj; yinej++)
                                                        {
                                                            if
(dugum[yinei, yinej] != 0)
                                                            {
boshucresay += 1;
                                                            }
                                                        }
                                                        if (yinesay ==
boshucresay)
                                                            gec = 1;
                                                    }
                                                    boshucresay = 0;
                                                    int ii = Grdi;
                                                    while (gec == 0 && ii
!= 13)
                                                    {
                                                        TEKRAR = TEKRAR +
2;
                                                        ii += 1;
                                                        for (int yinej =
SONSTN; yinej <= Grdj; yinej++)
                                                        {
                                                            gec = 1;
                                                            if (dugum[ii -
1, yinej] == 0 && dugum[ii, yinej] == 0)
                                                            {
                                                                gec = 0;
                                                            }
                                                        }
                                                    }
                                                    sayac += 1;
                                                    atama = 1;
                                                }
                                            }
                                        }
                                    }
                                    if (ENIYITEKRAR > TEKRAR)
                                    {
                                        ENIYISATIR = DONGUt;
                                        ENIYISUTUN = DONGUs;
                                        ENIYITEKRAR = TEKRAR;
                                    }
                                }
```

```
                }
            }

            sonrobot = 1;
            sayac = 1;
            for (int si = 0; si < 14; si++)
                for (int sj = 0; sj < 12; sj++)
                    dugum[si, sj] = yedekdugum[si, sj];
            dugum[ENIYISATIR, ENIYISUTUN] = 1;
            for (int t = 0; t < 14 * 12; t++)
            {
                for (int s = 0; s < 14 * 12; s++)
                {
                    atama = 0;
                    for (int Grdi = 0; Grdi < 14; Grdi++)
                        for (int Grdj = 0; Grdj < 12; Grdj++)
                        {
                            if (dugum[Grdi, Grdj] == 100 && atama ==
0)
                            {
                                for (int rbt = 1; rbt <= robotsay;
rbt++)
                                {
                                    if (Convert.ToDecimal(say *
(decimal)(rbt - 1)) < (decimal)sayac && (decimal)sayac <
Convert.ToDecimal(say * (decimal)rbt + (decimal)0.5))
                                    {
                                        if ((rbt != 1 &&
Convert.ToDecimal(say * (decimal)(rbt - 2)) < (decimal)(sayac - 1) &&
(decimal)(sayac - 1) <= Convert.ToDecimal(say * (decimal)(rbt-1) +
(decimal)0.5)) || (Grdi != 0 && dugum[Grdi - 1, Grdj] == rbt) || (Grdj
!= 0 && dugum[Grdi, Grdj - 1] == rbt) || (Grdi != 13 && dugum[Grdi +
1, Grdj] == rbt) || (Grdj != 11 && dugum[Grdi, Grdj + 1] == rbt))
                                        {
                                            dugum[Grdi, Grdj] = rbt;
                                            sonrobot = rbt;
                                            sayac += 1;
                                            atama = 1;
                                        }
                                    }
                                }
                                if (sayac > (say * robotsay))
                                {
                                    dugum[Grdi, Grdj] =
Convert.ToInt32(robotsay);
                                }
                            }
                        }
                }
                atama = 0;
                if (sayac != ((14 * 12) - engelsay))
                {
                    engel = 100;
                    for (int Grdi = 0; Grdi < 14; Grdi++)
                        for (int Grdj = 0; Grdj < 12; Grdj++)
                        {
                            if (dugum[Grdi, Grdj] == 100 && atama ==
0)
```

```
                        {
                            dugum[Grdi, Grdj] = sonrobot;
                            sayac += 1;
                            atama = 1;
                        }
                    }
                }
            }

            #endregion

            Graphics graph = this.CreateGraphics();

            for (int gX = 0; gX < 12; gX++)
                for (int gY = 0; gY < 14; gY++)
                    graph.DrawRectangle(new Pen(Color.Black, 1), new
System.Drawing.Rectangle((gX * 30), (gY * 30), 30, 30));

            #region İlk değer ataması yapma
            Degisken Deg = new Degisken { };
            int i = 0, j = 0, ilkI = 0, sonI = 0, ilkJ = 0, sonJ = 0,
indi = 0, indj = 0, ilk = 0, yolindis = 0, ilkyol = 0, ilkgeridon=0; /
            for (int a = 0; a < 14; a++)
                for (int b = 0; b < 12; b++)
                {
                    Deg.y[a, b] = 0;
                    Deg.t[a, b] = 0;
                    Deg.v[a, b] = 0;
                }
            for (int c = 0; c < 14; c++)
                for (int d = 0; d < 12; d++)
                    for (int f = 0; f < 14; f++)
                        for (int g = 0; g < 12; g++)
                            Deg.z[c, d, f, g] = 0;
            #endregion

            #region Kapsar Ağaç Oluşturma
            for (int x = 1; x <= robotsay; x++)
            {
                yolindis = 0;
                for (indi = 0; indi < 14; indi++)
                    for (indj = 0; indj < 12; indj++)
                    {
                        if (Deg.y[indi, indj] == 0 && dugum[indi,
indj] == x)
                        {
                            i = indi;
                            j = indj;
                            ilk = 0;
                            yolindis +=1;
                            ilkyol = 0;
                            yol = 1;
                            ilkgeridon=0;
                            maxX = 0;
                            maxY = 0;
                            maxt = 0;
                            for (int s = 0; s < Deg.HucreSayisi * 2;
s++)
```

```
                                {
                                    Deg.v[i, j] = yolindis;
                                    #region Yol tıkandığında geri dönme
                                    if ((i != 0 || j != 0) && yol == 0)
                                    {
                                        if (ilkgeridon == 1)
                                        {
                                            if (ilkI == sonI && ilkJ + 1
== sonJ)
                                            {
                                                if (i != 13 && dugum[(i +
1), j] == x && Deg.z[(i + 1), j, i, j] == 1)
                                                {
                                                    ilkI = i;
                                                    i = i + 1;
                                                    sonI = i;
                                                    ilkJ = j;
                                                    sonJ = j;
                                                }
                                                else if (j != 11 &&
dugum[i, (j + 1)] == x && Deg.z[i, (j + 1), i, j ] == 1)
                                                {
                                                    ilkJ = j;
                                                    j = j + 1;
                                                    sonJ = j;
                                                    ilkI = i;
                                                    sonI = i;
                                                }
                                                else if (i != 0 &&
dugum[(i - 1), j] == x && Deg.z[(i - 1), j, i, j] == 1)
                                                {
                                                    ilkI = i;
                                                    i = i - 1;
                                                    sonI = i;
                                                    ilkJ = j;
                                                    sonJ = j;
                                                }
                                            }

                                            else if (ilkI + 1 == sonI &&
ilkJ == sonJ)
                                            {
                                                if (j != 0 && dugum[i, (j
- 1)] == x && Deg.z[i, (j - 1), i, j] == 1)
                                                {
                                                    ilkJ = j;
                                                    j = j - 1;
                                                    sonJ = j;
                                                    ilkI = i;
                                                    sonI = i;
                                                }
                                                else if (i != 13 &&
dugum[(i + 1), j] == x && Deg.z[(i + 1), j, i, j] == 1)
                                                {
                                                    ilkI = i;
                                                    i = i + 1;
                                                    sonI = i;
                                                    ilkJ = j;
```

```
                                                sonJ = j;
                                            }
                                            else if (j != 11 &&
dugum[i, (j + 1)] == x && Deg.z[i, (j + 1), i, j] == 1)
                                            {
                                                ilkJ = j;
                                                j = j + 1;
                                                sonJ = j;
                                                ilkI = i;
                                                sonI = i;
                                            }
                                        }

                                        else if (ilkI == sonI && ilkJ
- 1 == sonJ)
                                        {
                                            if (i != 0 && dugum[(i -
1), j] == x && Deg.z[(i - 1), j, i, j] == 1)
{
                                                ilkI = i;
                                                i = i - 1;
                                                sonI = i;
                                                ilkJ = j;
                                                sonJ = j;
                                            }
                                            else if (j != 0 &&
dugum[i, (j - 1)] == x && Deg.z[i, (j - 1), i, j] == 1)
                                            {
                                                ilkJ = j;
                                                j = j - 1;
                                                sonJ = j;
                                                ilkI = i;
                                                sonI = i;
                                            }
                                            else if (i != 13 &&
dugum[(i + 1), j] == x && Deg.z[(i + 1), j, i, j] == 1)
                                            {
                                                ilkI = i;
                                                i = i + 1;
                                                sonI = i;
                                                ilkJ = j;
                                                sonJ = j;
                                            }
                                        }

                                        else if (ilkI - 1 == sonI &&
ilkJ == sonJ)
                                        {
                                            if (j != 11 && dugum[i, (j
+ 1)] == x && Deg.z[i, (j + 1), i, j] == 1)
                                            {
                                                ilkJ = j;
                                                j = j + 1;
                                                sonJ = j;
                                                ilkI = i;
                                                sonI = i;
                                            }
```

```
                                            else if (i != 0 &&
dugum[(i - 1), j] == x && Deg.z[(i - 1), j, i, j] == 1)
                                            {
                                                ilkI = i;
                                                i = i - 1;
                                                sonI = i;
                                                ilkJ = j;
                                                sonJ = j;
                                            }
                                            else if (j != 0 &&
dugum[i, (j - 1)] == x && Deg.z[i, (j - 1), i, j] == 1)
                                            {
                                                ilkJ = j;
                                                j = j - 1;
                                                sonJ = j;
                                                ilkI = i;
                                                sonI = i;
                                            }
                                        }
                                    }
                                    if (ilkgeridon == 0 && j != 11 &&
dugum[i, (j + 1)] == x && Deg.z[i, (j + 1), i, j] == 1)
                                    {
                                        ilkJ = j;
                                        j = j + 1;
                                        sonJ = j;
                                        ilkI = i;
                                        sonI = i;
                                        ilkgeridon = 1;
                                    }
                                    else if (ilkgeridon == 0 && j != 0
&& dugum[i, (j - 1)] == x && Deg.z[i, (j - 1), i, j] == 1)
                                    {
                                        ilkJ = j;
                                        j = j - 1;
                                        sonJ = j;
                                        ilkI = i;
                                        sonI = i;
                                        ilkgeridon = 1;
                                    }
                                    else if (ilkgeridon == 0 && i != 0
&& dugum[(i - 1), j] == x && Deg.z[(i - 1), j, i, j] == 1)
                                    {
                                        ilkI = i;
                                        i = i - 1;
                                        sonI = i;
                                        ilkJ = j;
                                        sonJ = j;
                                        ilkgeridon = 1;
                                    }
                                    else if (ilkgeridon == 0 && i !=
13 && dugum[(i + 1), j] == x && Deg.z[(i + 1), j, i, j] == 1)
                                    {
                                        ilkI = i;
                                        i = i + 1;
                                        sonI = i;
                                        ilkJ = j;
                                        sonJ = j;
```

```
                                        ilkgeridon = 1;
                                    }
                                }
                                #endregion
                                yol = 0;
                                #region Kapsar ağaç yolu oluşturma
                                if ((i == 0 && j == 0) || ilk == 0 ||
(yolindis>1 && ilkyol==0))
                                {
                                    Deg.y[i, j] = 1;
                                    Deg.t[i, j] += 1;
                                    #region Aynı robota ait kapsar
ağaçları birleştirme
                                    if ((i != 0 || j != 0) && yolindis
> 1 && ilkyol == 0)
                                    {
                                        if (j != 11 && dugum[i, (j +
1)] == x && Deg.y[i, j + 1] != 0 && Deg.z[i, j, i, (j+1)] != 1 &&
Deg.z[i, (j+1),i,j] !=1 && Deg.v[i,j] != Deg.v[i, (j+1)])
                                        {
                                            maxX = i;
                                            maxY = j + 1;
                                            maxt = Deg.t[i, (j + 1)];
                                        }
                                        if (i != 0 && dugum[(i - 1),
j] == x && Deg.y[(i - 1), j] != 0 && Deg.z[i, j, (i - 1), j] != 1 &&
Deg.z[(i - 1), j, i, j] != 1 && Deg.v[i, j] != Deg.v[(i-1), j])
                                        {
                                            if (maxt != 0)
                                            {
                                                if (Deg.t[maxX, maxY]
< Deg.t[(i - 1), j])
                                                {
                                                    maxX = i - 1;
                                                    maxY = j;
                                                    maxt = Deg.t[(i -
1), j];
                                                }
                                            }
                                            else
                                            {
                                                maxX = i - 1;
                                                maxY = j ;
                                                maxt = Deg.t[(i - 1),
j];
                                            }
                                        }
                                        if (j != 0 && dugum[i, (j -
1)] == x && Deg.y[i, (j - 1)] != 0 && Deg.z[i, j, i, (j - 1)] != 1 &&
Deg.z[i, (j - 1), i, j] != 1 && Deg.v[i, j] != Deg.v[i, (j - 1)])
                                        {
                                            if (maxt != 0)
                                            {
                                                if (Deg.t[maxX, maxY]
< Deg.t[i, (j - 1)])
                                                {
                                                    maxX = i;
                                                    maxY = j - 1;
```

```
                                                maxt = Deg.t[i, (j
- 1)];
                                            }
                                        }
                                        else
                                        {
                                            maxX = i;
                                            maxY = j - 1;
                                            maxt = Deg.t[i, (j -
1)];
                                        }
                                    }
                                    if (maxt != 0)
                                    {
                                        Deg.z[maxX, maxY, i, j] =
1;
                                        ilkyol = 1;
                                    }
                                }
                                #endregion
                                if ((i == 0 && j == 0) || ilk ==
0)
                                {
                                    if (j != 11 && dugum[i, (j +
1)] == x && Deg.y[i, j + 1] == 0)
                                    {
                                        Deg.z[i, j, i, (j + 1)] =
1;
                                        max += 1;
                                        Deg.t[i, (j + 1)] = max;
                                        Deg.y[i, j + 1] = 1;
                                        ilkI = i;
                                        sonI = i;
                                        ilkJ = j;
                                        j = j + 1;
                                        sonJ = j;
                                        yol = 1;
                                        ilk = 1;
                                    }
                                    else if (i != 13 && dugum[(i +
1), j] == x && Deg.y[(i + 1), j] == 0)
                                    {
                                        Deg.z[i, j, (i + 1), j] =
1;
                                        max += 1;
                                        Deg.t[(i + 1), j] = max;
                                        Deg.y[(i + 1), j] = 1;
                                        ilkI = i;
                                        ilkJ = j;
                                        sonJ = j;
                                        i = i + 1;
                                        sonI = i;
                                        yol = 1;
                                        ilk = 1;
                                    }
                                    else if (j != 0 && dugum[i, (j
- 1)] == x && Deg.y[i, (j - 1)] == 0)
                                    {
```

```
                                                Deg.z[i, j, i, (j - 1)] =
1;
                                                max += 1;
                                                Deg.t[i, (j - 1)] = max;
                                                Deg.y[i, (j - 1)] = 1;
                                                ilkI = i;
                                                sonI = i;
                                                ilkJ = j;
                                                j = j - 1;
                                                sonJ = j;
                                                yol = 1;
                                                ilk = 1;
                                            }
                                            else if (i != 0 && dugum[(i -
1), j] == x && Deg.y[(i - 1), j] == 0)
                                            {
                                                Deg.z[i, j, (i - 1), j] =
1;
                                                max += 1;
                                                Deg.t[(i - 1), j] = max;
                                                Deg.y[(i - 1), j] = 1;
                                                ilkI = i;
                                                ilkJ = j;
                                                sonJ = j;
                                                i = i - 1;
                                                sonI = i;
                                                yol = 1;
                                                ilk = 1;
                                            }
                                        }
                                    }
                                    else if (ilkI == sonI && ilkJ + 1 ==
sonJ)
                                    {
                                        if (i != 0 && dugum[(i - 1), j] ==
x && Deg.y[(i - 1), j] == 0)
                                        {
                                            Deg.z[i, j, (i - 1), j] = 1;
                                            max += 1;
                                            Deg.t[(i - 1), j] = max;
                                            Deg.y[(i - 1), j] = 1;
                                            Deg.y[i, j] = 2;
                                            ilkI = i;
                                            i = i - 1;
                                            sonI = i;
                                            ilkJ = j;
                                            sonJ = j;
                                            yol = 1;
                                        }
                                        else if (j != 11 && dugum[i, (j +
1)] == x && Deg.y[i, (j + 1)] == 0)
                                        {
                                            Deg.z[i, j, i, (j + 1)] = 1;
                                            max += 1;
                                            Deg.t[i, (j + 1)] = max;
                                            Deg.y[i, (j + 1)] = 1;
                                            Deg.y[i, j] = 1;
                                            ilkJ = j;
```

```
                    j = j + 1;
                    sonJ = j;
                    ilkI = i;
                    sonI = i;
                    yol = 1;
                }
                else if (i != 13 && dugum[(i + 1),
j] == x && Deg.y[(i + 1), j] == 0)
                {
                    Deg.z[i, j, (i + 1), j] = 1;
                    max += 1;
                    Deg.t[(i + 1), j] = max;
                    Deg.y[(i + 1), j] = 1;
                    Deg.y[i, j] = 2;
                    ilkI = i;
                    i = i + 1;
                    sonI = i;
                    ilkJ = j;
                    sonJ = j;
                    yol = 1;
                }
            }

            else if (ilkI + 1 == sonI && ilkJ ==
sonJ)
            {
                if (j != 11 && dugum[i, (j + 1)]
== x && Deg.y[i, (j + 1)] == 0)
                {
                    Deg.z[i, j, i, (j + 1)] = 1;
                    max += 1;
                    Deg.t[i, (j + 1)] = max;
                    Deg.y[i, (j + 1)] = 1;
                    Deg.y[i, j] = 2;
                    ilkJ = j;
                    j = j + 1;
                    sonJ = j;
                    ilkI = i;
                    sonI = i;
                    yol = 1;
                }
                else if (i != 13 && dugum[(i + 1),
j] == x && Deg.y[(i + 1), j] == 0)
                {
                    Deg.z[i, j, (i + 1), j] = 1;
                    max += 1;
                    Deg.t[(i + 1), j] = max;
                    Deg.y[(i + 1), j] = 1;
                    Deg.y[i, j] = 1;
                    ilkI = i;
                    i = i + 1;
                    sonI = i;
                    ilkJ = j;
                    sonJ = j;
                    yol = 1;
                }
                else if (j != 0 && dugum[i, (j -
1)] == x && Deg.y[i, (j - 1)] == 0)
```

```
                {
                    Deg.z[i, j, i, (j - 1)] = 1;
                    max += 1;
                    Deg.t[i, (j - 1)] = max;
                    Deg.y[i, (j - 1)] = 1;
                    Deg.y[i, j] = 2;
                    ilkJ = j;
                    j = j - 1;
                    sonJ = j;
                    ilkI = i;
                    sonI = i;
                    yol = 1;
                }
            }

            else if (ilkI == sonI && ilkJ - 1 ==
sonJ)
            {
                if (i != 13 && dugum[(i + 1), j]
== x && Deg.y[(i + 1), j] == 0)
                {
                    Deg.z[i, j, (i + 1), j] = 1;
                    max += 1;
                    Deg.t[(i + 1), j] = max;
                    Deg.y[(i + 1), j] = 1;
                    Deg.y[i, j] = 2;
                    ilkI = i;
                    i = i + 1;
                    sonI = i;
                    ilkJ = j;
                    sonJ = j;
                    yol = 1;
                }
                else if (j != 0 && dugum[i, (j -
1)] == x && Deg.y[i, (j - 1)] == 0)
                {
                    Deg.z[i, j, i, (j - 1)] = 1;
                    max += 1;
                    Deg.t[i, (j - 1)] = max;
                    Deg.y[i, (j - 1)] = 1;
                    Deg.y[i, j] = 1;
                    ilkJ = j;
                    j = j - 1;
                    sonJ = j;
                    ilkI = i;
                    sonI = i;
                    yol = 1;
                }
                else if (i != 0 && dugum[(i - 1),
j] == x && Deg.y[(i - 1), j] == 0)
                {
                    Deg.z[i, j, (i - 1), j] = 1;
                    max += 1;
                    Deg.t[(i - 1), j] = max;
                    Deg.y[(i - 1), j] = 1;
                    Deg.y[i, j] = 2;
                    ilkI = i;
                    i = i - 1;
```

```
                    sonI = i;
                    ilkJ = j;
                    sonJ = j;
                    yol = 1;
                }
            }

            else if (ilkI - 1 == sonI && ilkJ ==
sonJ)
            {
                if (j != 0 && dugum[i, (j - 1)] ==
x && Deg.y[i, (j - 1)] == 0)
                {
                    Deg.z[i, j, i, (j - 1)] = 1;
                    max += 1;
                    Deg.t[i, (j - 1)] = max;
                    Deg.y[i, (j - 1)] = 1;
                    Deg.y[i, j] = 2;
                    ilkJ = j;
                    j = j - 1;
                    sonJ = j;
                    ilkI = i;
                    sonI = i;
                    yol = 1;
                }
                else if (i != 0 && dugum[(i - 1),
j] == x && Deg.y[(i - 1), j] == 0)
                {
                    Deg.z[i, j, (i - 1), j] = 1;
                    max += 1;
                    Deg.t[(i - 1), j] = max;
                    Deg.y[(i - 1), j] = 1;
                    Deg.y[i, j] = 1;
                    ilkI = i;
                    i = i - 1;
                    sonI = i;
                    ilkJ = j;
                    sonJ = j;
                    yol = 1;
                }
                else if (j != 11 && dugum[i, (j +
1)] == x && Deg.y[i, (j + 1)] == 0)
                {
                    Deg.z[i, j, i, (j + 1)] = 1;
                    max += 1;
                    Deg.t[i, (j + 1)] = max;
                    Deg.y[i, (j + 1)] = 1;
                    Deg.y[i, j] = 2;
                    ilkJ = j;
                    j = j + 1;
                    sonJ = j;
                    ilkI = i;
                    sonI = i;
                    yol = 1;
                }
            }
            #endregion
            ilk = 1;
```

```
                            }
                        }
                    }
                }
                #endregion

                #region Kapsanmadan kalmış tek hücreleri yola dahil etme
                for (int a = 0; a < 14; a++)
                    for (int b = 0; b < 12; b++)
                    {
                        maxt = 0;
                        if (Deg.y[a, b] == 0 && dugum[a, b] != 0)
                        {
                            if (b != 11 && dugum[a, (b + 1)] == dugum[a,
b] && Deg.y[a, b + 1] != 0)
                            {
                                maxX = a;
                                maxY = b + 1;
                                maxt = Deg.t[a, (b + 1)];
                            }
                            if (a != 0 && dugum[(a - 1), b] == dugum[a, b]
&& Deg.y[(a - 1), b] != 0)
                            {
                                if (maxt != 0)
                                {
                                    if (Deg.t[maxX, maxY] < Deg.t[(a - 1),
b])
                                    {
                                        maxX = a - 1;
                                        maxY = b;
                                        maxt = Deg.t[(a - 1), b];
                                    }
                                }
                                else
                                {
                                    maxX = a - 1;
                                    maxY = b;
                                    maxt = Deg.t[(a - 1), b];
                                }
                            }
                            if (b != 0 && dugum[a, (b - 1)] == dugum[a, b]
&& Deg.y[a, (b - 1)] != 0)
                            {
                                if (maxt != 0)
                                {
                                    if (Deg.t[maxX, maxY] < Deg.t[a, (b -
1)])
                                    {
                                        maxX = a;
                                        maxY = b - 1;
                                        maxt = Deg.t[a, (b - 1)];
                                    }
                                }
                                else
                                {
                                    maxX = a;
                                    maxY = b - 1;
                                    maxt = Deg.t[a, (b - 1)];
```

```
                            }
                        }
                        if (maxt != 0)
                        {
                            Deg.z[maxX, maxY, a, b] = 1;
                        }
                    }
                }
            #endregion

            #region Çözümü çizdirme
            for (int c = 0; c < 14; c++)
                for (int d = 0; d < 12; d++)
                {
                    if (dugum[c, d] == 0)
                    {
                        for (int a = 0; a < 16; a++ )
                            graph.DrawLine(new Pen(Color.DimGray, 2),
new System.Drawing.Point((a * 2) + (30 * d), (30 * c)), new
System.Drawing.Point((a * 2) + (30 * d), 30 + (30 * c)));
                    }

                    for (int f = 0; f < 14; f++)
                        for (int g = 0; g < 12; g++)
                        {
                            if (Deg.z[c, d, f, g] == 1)
                            {
                                if (dugum[c, d] == 1)
                                    graph.DrawLine(new Pen(Color.Red,
2), new System.Drawing.Point(15 + (30 * d), 15 + (30 * c)), new
System.Drawing.Point(15 + (30 * g), 15 + (30 * f)));
                                else if (dugum[c, d] == 2)
                                    graph.DrawLine(new Pen(Color.Blue,
2), new System.Drawing.Point(15 + (30 * d), 15 + (30 * c)), new
System.Drawing.Point(15 + (30 * g), 15 + (30 * f)));
                                else if (dugum[c, d] == 3)
                                    graph.DrawLine(new
Pen(Color.Green, 2), new System.Drawing.Point(15 + (30 * d), 15 + (30
* c)), new System.Drawing.Point(15 + (30 * g), 15 + (30 * f)));
                                else if (dugum[c, d] == 4)
                                    graph.DrawLine(new
Pen(Color.Orange, 2), new System.Drawing.Point(15 + (30 * d), 15 + (30
* c)), new System.Drawing.Point(15 + (30 * g), 15 + (30 * f)));
                                else if (dugum[c, d] == 5)
                                    graph.DrawLine(new
Pen(Color.Turquoise, 2), new System.Drawing.Point(15 + (30 * d), 15 +
(30 * c)), new System.Drawing.Point(15 + (30 * g), 15 + (30 * f)));
                                else if (dugum[c, d] == 6)
                                    graph.DrawLine(new
Pen(Color.YellowGreen, 2), new System.Drawing.Point(15 + (30 * d), 15
+ (30 * c)), new System.Drawing.Point(15 + (30 * g), 15 + (30 * f)));
                                else if (dugum[c, d] == 7)
                                    graph.DrawLine(new Pen(Color.Navy,
2), new System.Drawing.Point(15 + (30 * d), 15 + (30 * c)), new
System.Drawing.Point(15 + (30 * g), 15 + (30 * f)));
                                else if (dugum[c, d] == 8)
```

```
                                    graph.DrawLine(new Pen(Color.Peru,
2), new System.Drawing.Point(15 + (30 * d), 15 + (30 * c)), new
System.Drawing.Point(15 + (30 * g), 15 + (30 * f)));
                                else if (dugum[c, d] == 9)
                                    graph.DrawLine(new Pen(Color.Teal,
2), new System.Drawing.Point(15 + (30 * d), 15 + (30 * c)), new
System.Drawing.Point(15 + (30 * g), 15 + (30 * f)));
                                else if (dugum[c, d] == 10)
                                    graph.DrawLine(new
Pen(Color.Violet, 2), new System.Drawing.Point(15 + (30 * d), 15 + (30
* c)), new System.Drawing.Point(15 + (30 * g), 15 + (30 * f)));
                                else
                                    MessageBox.Show("Robot sayısı en
fazla 10 olmalıdır !!!", "Robot Sayısında Hata",MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);
                            }
                        }
                }

            #endregion
        }

        private void btnGridGetir_Click(object sender, EventArgs e)
        {
            sGrdDugum.Visible = true;
            dataGridView1.Visible = false;

            sGrdDugum.ColumnCount = 12;
            sGrdDugum.RowCount = 14;
            for (int q = 0; q < sGrdDugum.ColumnCount; q++)
                sGrdDugum.FormatColumn(q, "-" + (q + 1).ToString() +
"-", "center", 30, false);

            for (int t = 0; t < sGrdDugum.RowCount; t++)
                for (int z = 0; z < sGrdDugum.ColumnCount; z++)
                    sGrdDugum[t, z] = " ";
        }

        private void btnVeriYukle_Click(object sender, EventArgs e)
        {
            sGrdDugum.Visible = false;
            dataGridView1.Visible = true;

            OpenFileDialog openFileDialog1 = new OpenFileDialog();

            openFileDialog1.InitialDirectory = "c:\\";
            openFileDialog1.Filter = "xls files (*.xls)|*.xls|All
files (*.*)|*.*";
            openFileDialog1.FilterIndex = 2;
            openFileDialog1.RestoreDirectory = true;
            openFileDialog1.Title = "Lütfen yerleşim bilgilerinin
olduğu dosyayı seçiniz:";

            if (openFileDialog1.ShowDialog() == DialogResult.OK)
            {
                if (File.Exists(openFileDialog1.FileName))
                {
                    dataGridView1.ColumnCount = 12;
```

```
                    dataGridView1.ColumnHeadersVisible = true;
                    for (int j = 0; j <= 11; j++)
                    {
                        DataGridViewColumn column =
dataGridView1.Columns[j];
                        column.Width = 28;
                    }
                    dataGridView1.Columns[0].Name = "B";
                    dataGridView1.Columns[1].Name = "C";
                    dataGridView1.Columns[2].Name = "D";
                    dataGridView1.Columns[3].Name = "E";
                    dataGridView1.Columns[4].Name = "F";
                    dataGridView1.Columns[5].Name = "G";
                    dataGridView1.Columns[6].Name = "H";
                    dataGridView1.Columns[7].Name = "I";
                    dataGridView1.Columns[8].Name = "J";
                    dataGridView1.Columns[9].Name = "K";
                    dataGridView1.Columns[10].Name = "L";
                    dataGridView1.Columns[11].Name = "M";
                    Excel.Workbook theWorkbook =
ExcelNesnesi.Workbooks.Open(
                        openFileDialog1.FileName,
                        Type.Missing,
                        Type.Missing,
                        Type.Missing,
                        Type.Missing,
                        Type.Missing,
                        Type.Missing,
                        Type.Missing,
                        Type.Missing,
                        Type.Missing,
                        Type.Missing,
                        Type.Missing,
                        Type.Missing,
                        Type.Missing);
                    Excel.Sheets sheets = theWorkbook.Worksheets;
                    Excel.Worksheet worksheet =
(Excel.Worksheet)sheets.get_Item(1);

                    for (int i = 2; i <= 15; i++)
                    {
                        Excel.Range range = worksheet.get_Range("B" +
i.ToString(), "M" + i.ToString());
                        System.Array myvalues =
(System.Array)range.Cells.Value2;
                        string[] strArray =
ConvertToStringArray(myvalues);
                        dataGridView1.Rows.Add(strArray);

                    }
                    theWorkbook.Close(false, null, null);
                    ExcelNesnesi.Workbooks.Close();

                }
            }
        }
```

```
        string[] ConvertToStringArray(System.Array values)
        {
            string[] theArray = new string[values.Length];
            for (int i = 1; i <= values.Length; i++)
            {
                if (values.GetValue(1, i) == null)
                    theArray[i - 1] = "";
                else
                    theArray[i - 1] = (string)values.GetValue(1,
i).ToString();
            }
            return theArray;
        }

        private void btnTemizle_Click(object sender, EventArgs e)
        {
            Graphics graph = this.CreateGraphics();
            graph.DrawRectangle(new Pen(Color.Black, 2), 1, 1, 359,
419);
            graph.FillRectangle(new SolidBrush(Color.Linen), 1, 1,
359, 419);
        }


    }
}
```