Karadeniz Teknik Üniversitesi
Bilgisayar Mühendisliği Bölümü
Optimizasyon

İNSANSIZ ARAÇLARDA YOL PLANLANMASI

Öğretim	1. Öğretim ☒	2. Öğretim ☐
	Ad ve Soyad	
348337	Fatma SÖZER	
Prof.Dr	Vasif Nabiyev	

İNSANSIZ ARAÇLARDA YOL PLANLANMASI

İnsansız hava ve kara araçları için engellerden ve tehlikeli bölgelerden sakınarak güvenli bir şekilde en kısa yoldan hedefe ulaşması büyük önem taşımaktadır. Hızlı, maliyeti düşük ve güvenli olan olan algoritmaları aşağıda sıraladık.

1.RASTGELE AĞAÇLAR YÖNTEMİ

Verilen başlangıç noktasından hedefe kadar rastgele noktalar seçerek dallanmalar ile hedef noktasına ulaşır. Rastgele seçilecek noktaların belirli bir sınır değeri vardır. Seçilecek her yeni nokta kendinden önceki noktalar gözardı edilerek sadece başlangıç noktasına göre yeri belirlenir. Seçilen nokta ile önceki nokta arasında bağlantı oluşur. Bağlantı oluşan yerde eğer engel olursa seçilen yeni noktadan vazgeçilir. Yapılan işlemler seçilen nokta hedef noktasına en yakın ya da hedef noktası olana kadar tekrarlanır. Birbirinden farklı olarak kurulan sınıflama ve regresyon karar ağaçları (CART) karar ormanı topluluğunu oluşturur. Sınıflandırma için ağaçlar; her yaprak düğümü sadece bir sınıfın üyelerini içerecek şekilde oluşturulurlar. Regresyon için ise; yaprak düğümde az sayıda birim kalana kadar ağaçlar bölünmeye devam ederler.

Eşsiz bir tahmin geçerliliği ve model yorumlanabilirliği sağlar bundan dolayı daha iyi genellemeler sunar ve geçerli tahminlerde bulunur. Mümkün olduğunca birbirinden farklı ağaçlar oluşturarak da düşük korelasyon yapısında bir topluluk elde edilir.

Faydaları :

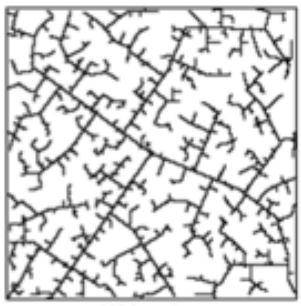
- Oldukça geçerli bir uygulamadır.
- Çalışma zamanı kısa.
- Sayıca fazla değişken ve sınıflarda veya dengesiz dağılıma sahip veri setlerini kullarak sonuç verir.
- Ağaç sayısı fazlalaştıkça yanılma payı azalır.

Rastgele ağaçlar yöntemini inceleyelim.

Algoritma kullanıcı tarafından iki sayı alır.

M: En iyi bölünmeyi belirlemek için her bir düğümde kullanılan değişkenlerin sayısı.

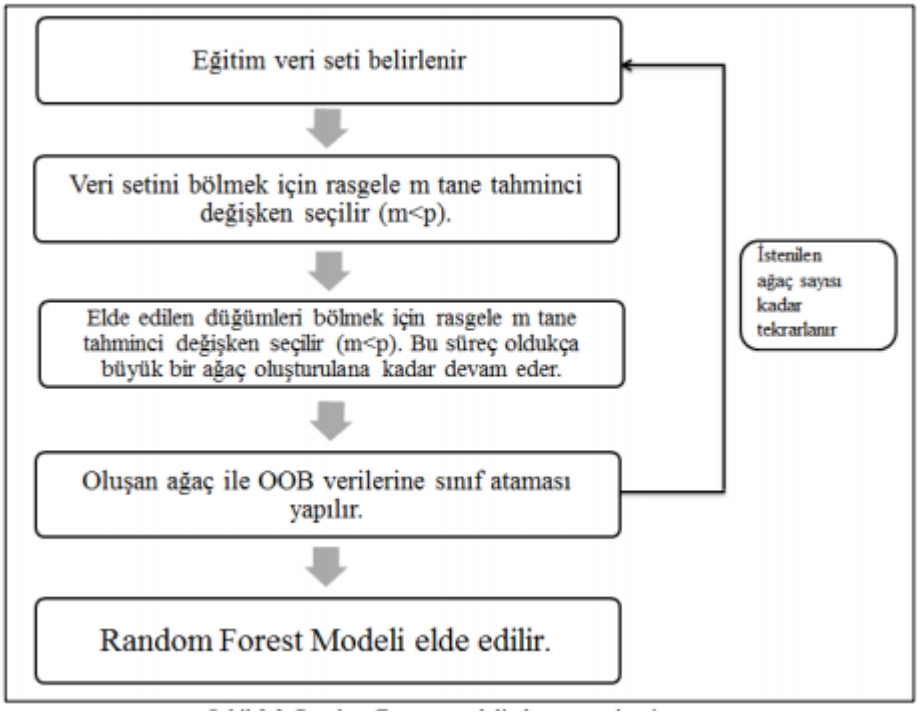
N: Geliştirilecek ağaç sayısı.



Orijinal veri setindeki gözlem sayısı B'dir. Alının değerlerin 2/3'ünden önyükleme örnekleri (eğitici veri seti – ağaç) oluşturulur. Geri kalan 1/3'lük kısmı hataları test etmek için(out of bag) kullanılır. Her yaprak düğümü sadece bir sınıfın üyelerini içerecek şekilde oluşturulurlar.

Algoritmayı inceleyelim

- Bootstrap yöntemi ile veri setindeki bilgiler seçilir, eğitim veri seti (inBag) ve test veri seti (OOB) olarak ikiye ayrılır.
- Eğitim veri setiyle olduğunca geniş kara ağacı oluşturulur. Ağaç oluşumunda p tane tahminci değişkenden M tanesi rastgele seçilir. Ağacın çok büyük olması istenmediği için M < p olmalıdır. M tahminciden bilgi açısından en faydalı olan ayrılma gerçekleşir. Değişkenin hangi değerine ayrım olacağının belirlenmesi gini indeksi ile olur. İşlem her düğüm için yeni oluşturulacak dal kalmayıncaya kadar tekrar edilir.
- Her düğüme bir sınıf atanır. Test veri seti ağacın en tepesine konur ve set sayesinde her gözleme atanan sınıf kaydedilir.
- Bu aşamalar N defa tekrar edilir.
- Ağaç oluşumunda kullanılmayan gözlemler ile değerlendirme yapılır.
- Gözlemler, ağaç setleri üzerinden belirlenen bir oy çoğunluğu ile sınıf ataması yapılır.



Tablo modeli yukarıda verilmiştir.

RF yönteminde karar ormanı oluşturulurken belirlenmesi gereken 2 parametre vardır; her düğümde rasgele seçilecek olan değişken sayısı (M) ve oluşturulacak ağaç sayısıdır (N). RF bu parametrelerin seçiminde hassas bir yapı sergilemez. Breiman, bu parametrelerin seçimi için bazı önerilerde bulunmuştur. Breiman'a göre 500 adet ağaçtan oluşacak bir karar ormanı yeterli sayılabilir. Pek çok sınıflandırma problemi için her düğümde rasgele seçilecek olan değişken sayısı;

M = sqrt p

p : Veri setindeki tahminci değişkenleri sayısı

eşitliği ile hesaplanmaktadır. Regresyon ağaçlarında ise m parametresi m=p/3 olarak elde edilir.

İstatistiksel analizler uygulanmadan önce, yüksek boyutlu veri setini indirgemek için temel bileşenler analizi kullanılsa da, yöntem tahmin için önemli bilgileri yakalayamamaktadır. Bu durumda değişken önemliliğini direk algoritmadan gözlemlemek ve önemli değişkenler kullanılarak model kurmak daha çok tercih edilen bir durumdur. Değişken önemliliği rastgele ağaçlar yönteminin içindedir. Yöntem ile hesaplanma yapmadaki en önemli amaçlar; model performansını geliştirerek aşırı uyumu engellemek ve veri setini türeten sürecin altında yatan kavramı daha derinden anlamaktır.

Gini İdeksi

Gini indeksi bir düğüme atanmış örneklemin karışıklık ya da eşitsizlik seviyesini ölçer

$G_k = 2p (1 – p)$

p = k düğümünde yer alan pozitif gözlemlerin oranı

1-p = negatif gözlemlerin oranı

G_k = k düğümünde yer alan gini indeksi

Gini indeksi belirlendikten sonra test veri setlerimiz gini indeksi baz alınarak sınıflar belirlenir. Tüm bu adımlardan sonra en uygun sınıflandırma yapılmış olur. Gini indeksi büyüdükçe sınıf heterojenliği artar ve gini indeksi küçüldükçe sınıf homojenliği artar. GINI indeksi düşük olan dal başarılıdır. Her bir yaprak düğümünde bir sınıf kaldığında gini indeksi sıfırlanır ve ağaç dallanma işlemi sonlanır. N adet ağaç üretildiğinde N tane ağaçtan alınan bilgiler ile aday pikselin sınıfı belirlenir. N tane olan ağaçların tahminlerinden yola çıkarak yeni veriler elde edilir.

RF yönteminde v değişkeninin önem derecesi aşağıdaki sıralama ile bulunur. Öncelikle OOB gözlemleri ağaçtan aşağı bırakılır ve tahmin edilen değerler belirlenir. Daha sonra ise OOB' de yer alan diğer tahminci değişkenler sabit olmak koşulu ile v değişkenine ait gözlem değerleri rasgele karıştırılır.

Elde edilen yeni OOB veri seti ağaçtan aşağı bırakılır ve tahmin edilen değerler belirlenir. Bu işlem sonucunda her gözlem için iki tane tahmin değeri elde edilmiş olur. Orijinal OOB ile elde edilen doğru tahmin sayısından, değiştirilmiş OOB ile elde edilen doğru tahmin sayısı çıkartılarak bir fark elde edilir. Bu işlem tüm ormana uygulanarak ormandaki ağaç sayısı kadar fark elde edilir ve bu farkların ortalaması hesaplanır. Tüm ağaçların birbirinden bağımsız olduğu ve elde edilen fark değerlerinin normal dağıldığı varsayımı altında v değişkeni için z skor değeri hesaplanır. Bu skor değeri; farklar ortalamasının farkların standart hatasına oranlanması ile hesaplanır. Ağaçta yer alan her v değişkeni için skor değerler elde edilir. Elde edilen skor değerlerine göre değişkenlerin önemlilik dereceleri kıyaslanarak bir sıralama belirlenmiş olur.

Regresyon problemlerinde ağaçların derinliğinin ya da yaprak düğümlerde kalacak olan minimum gözlem sayısının kontrol edilmesi gereklidir. Regresyonda, safsızlık fonksiyonu i_R (t) kare hata kaybında tanımlanan yerel yeniden yerleşme tahmini

$i_R(j) = (1/N) \quad \Sigma_{j=1}^{N} (y_j - m)^2$

Impurity	Task	Formula	Description
Gini impurity	Classification	$\sum_{i=1}^{C} f_i(1-f_i)$	f_i is the frequency of label i at a node and C is the number of unique labels.
Entropy	Classification	$\sum_{i=1}^{C} -f_i \log(f_i)$	f_i is the frequency of label i at a node and C is the number of unique labels.
Variance / Mean Square Error (MSE)	Regression	$\frac{1}{N}\sum_{i=1}^{N}(y_i-\mu)^2$	y_i is label for an instance, N is the number of instances and μ is the mean given by $\frac{1}{N}\sum_{i=1}^{N} y_i$
Variance / Mean Absolute Error (MAE) (Scikit-learn only)	Regression	$\frac{1}{N}\sum_{i=1}^{N}\lvert y_i-\mu\rvert$	y_i is label for an instance, N is the number of instances and μ is the mean given by $\frac{1}{N}\sum_{i=1}^{N} y_i$

Not : Tabloda kullanılan bazı formüller verilmiştir.

Faydaları

- Regrasyon problemlerinin yanı sıra sınıflandırmayı çözmek için kullanılır.
- Hem kategorik hem de sürekli değişkenlerle iyi çalışır.

2. YAPAY ARI KOLONİSİ ALGORİTMASI

Doğayı taklit ederek geliştirilen algoritmada çözülmek istenen problemin olası çözümlerini, kaynaklardaki nektar miktarı ise çözümün kalitesini (amaç fonksiyon) ifade etmektedir. Algoritmada doğada ki gibi üç çeşit arı bulunmaktadır. Bunlar işçi, gözcü ve kaşif arılardır. Algoritma çalışma mantığı aşağıdaki gibidir.

İlk önce kaşif arılar yiyecek kaynaklarını rastgele aramaya başlar.

i = 1,2,…,SN. (SN aday çözüm sayısı)

j = 1,2,…,D. (optimize edilecek parametre sayısı)

rand(0,1) =0 ve 1 değerleri arasında rastgele sayı üretmektedir.

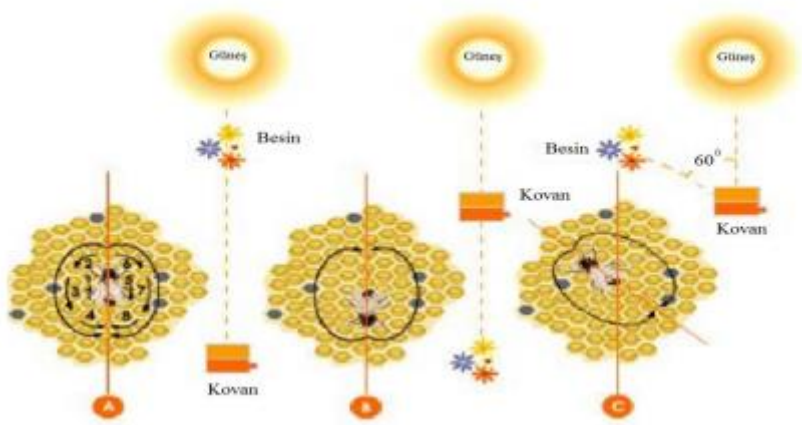
x^{max}, x^{min} = j parametresinin alt ve üst sınır değerlerini belirler.

x_{ij} = rastgele yiyecek kaynakları. Başlangıç nüfusu olarak üretilen bu kaynaklar aday çözüm kümesini ifade etmektedir.

$x_{ij} = x_j^{min} + rand(0,1)\,(x_j^{max} - x_j^{min})$

Bulunan kaynağında besin olup olmadığı geliştirilememe sayacı ile kontrol edilir. Bu kontrol parametresi önceden tanımlı olup “limit“ ismi ile anılmaktadır. Besin olup olamaması

arılar için büyük önem taşımaktadır. Kaşif arının her keşfe çıkışında limit değeri sıfırlanır. Algoritmanın durması için ya kullanıcı tarafından belirlenen kriter yada maksimum çevrim sayısı olması gerekir. Kaşif arılar işlerini tamamladıktan sonra görevi işçi arılara teslim ederler.



İşçi Arılar

İşçi arılar bulunan kaynağı ve araştırmaları sonucu yeni buldukları komşu kaynakları besin miktarları açısından değerlendirir. Hangi kaynağın besini daha iyi ise sadece onu aklında tutar diğerlerini siler. Bulunan kaynağın etrafında yeni kaynaklar araması için kullanılan denklem aşağıdaki gibidir.

x_{ij} = bulunan kaynak.

v_i = x_i için 0 ve 1 arasında rastgele seçilen j değeri değiştirilerek bulunan yeni kaynak.

o_{ij} = -1 ve 1 arasında rastgele olan sayı.

x_{kj} = rastgele üretilen kaynak üretilen.

$$v_{ij} = x_{ij} + o_{ij} (x_{ij} - x_{kj})$$

Bulunan yeni komşu kaynağın kalitesi ve önceki kaynağımızın kalitesi karşılaştırmamız gerekmektedir.

f_i = komşu kaynak kalitesi.

fit_i = bulunan kaynağın uygunluk değeri.

$$fit_i = \begin{cases} 1 / (1 + f_i) & , f_i \geq 0 \\ 1 + | f_i | & , f_i < 0 \end{cases}$$

Komşu değer ile mevcut değer arasında yapılan karşılaştırma ile mevcut değer daha kaliteli ise kaynak olarak hala mevcut kaynak kullanılır sayaç bir arttırılır ama komşu değer daha kaliteli ise yeni kaynağımız mevcut kaynak olup geliştirememe sayacı sıfırlanır.

Gözcü Arılar

Tüm bu araştırma işlemlerinden sonra gözcü arılar elde edilen tüm bilgileri toplar. Kaynaktaki nektar miktarı ile orantılı olasılıkla bir kaynak seçer. Her kaynak için aşağıdaki formüle göre,

$$P_i = fit_i \; / \; \sum_{i=1}^{sn} fit_i$$

P_i = i. çözümün olasılık değeri

fit_i = i. çözümün uygunluk değeri

Hesaplanan olasılık değeri [0,1] arasında ve rastgele üretilen değerden büyük ise gözcü arılar ;

$$v_{ij} = x_{ij} + o_{ij} (x_{ij} - x_{kj})$$

denklemi ile yeni çözüm ve çözümü kalitesi hesaplanır.

Kaşif Arılar

Yiyecek kaynağı, önceden belirlenmiş olan geliştirilememe sayacı “limit” eşik değerine ulaşmış ise, bu yiyecek kaynağı tükenmiş olarak kabul edilir ve bu noktadan sonra bu kaynaktaki görevli arı kâşif arıya dönüşür. Arı tükenen kaynak yerine denklem ile tekrar kaynak oluşturur.

$$x_{ij} = x_j^{min} + \text{rand}(0,1) (x_j^{max} - x_j^{min})$$

Rastgele başlangıç kaynakları oluştur, x_{ij} , $i = 1...SN$, $j = 1...D$

Her bir kaynağın amaç fonksiyonunu hesapla, f_i

iterasyon = 1

repeat

- $v_{ij} = x_{ij} + o_{ij} (x_{ij} - x_{kj})$

Denklemini kullanarak vi komşu kaynağını üret ve fi değerini hesapla

- *vi ile xi arasında açgözlü seçim işlemi uygula*

- $P_i = fit_i \; / \; \sum_{i=1}^{sn} fit_i$

Denklemini kullanarak kaynakların olasılıklarını belirle, pi - pi olasılığına göre xi kaynağını seç

- *xi komşuluğunda yeni bir vi kaynağını üret ve fi değerini hesapla*

- *vi ile xi arasında açgözlü seçim işlemi uygula*

- Tükenen kaynak var ise o kaynak için

$x_{ij} = x_j^{min} + rand\ (0,1)\ (x_j^{max} - x_j^{min})$

denklemi ile rastgele yeni bir kaynak üret - Şimdiye kadar bulunan en iyi çözümü sakla

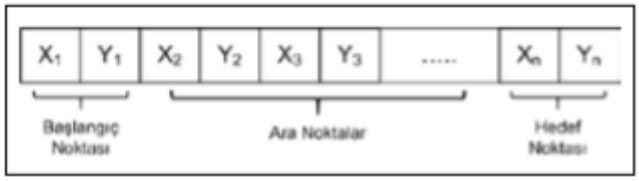
- iterasyon = iterasyon + 1;

until iterasyon = maksimum çevrim sayısı

Gerçeklenen tüm bu adımlar ile gerçekleştirememe sayacı limit değerine ulaştığında kullanılan yiyecek kaynağı bitmiş olur ve tekrar kaşif arıya görev gelir. Adımlar başa döner. Gerçeklenen adımlar maksimum çevrim sayısına ulaşıncaya kadar veya tanımlanan kriter olana kadar devam eder. Daha sonrasında algoritma sonlandırılır.

Yapay Arı Kolonisi Algoritması ile Rota Planlama

İnsansız araçların başlangıç ve hedef noktaları arasında geçilecek bağlantı koordinat noktalarının tespit edilip, o noktalar üzerinden geçerek hedefe varması hedeflenmektedir. Geçilecek bağlantı noktaları belirlenirken iki temel kriter dikkate alınmıştır. Bu kriterler; rota üzerindeki engellerin dikkate alınması ve rotanın en kısa uzunlukta olacak şekilde belirlenmesi. Başlangıç ve hedef arasındaki mesafeye ve engel sayısına göre değişen haritalarda sabit bağlantı noktaların kullanılması dezavantaj oluşturabilmektedir. Bu bakımdan çalışmada, rota üretimi sırasında başlangıç ve hedef koordinatları arasında üzerinden geçilecek olan bağlantı koordinat noktalarının sayısı dinamik olarak belirlenmektedir. Her bir bağlantı koordinat noktası için rastgele enlem ve boylam bilgisi üretilmektedir. Algoritmada kullanılan



Rota yapısı şekilde gösterilmektedir.

Amaç Fonksiyon

Rastgele rotaların üretiminden sonra her bir rotanın maliyeti (amaç fonksiyon) hesaplanmaktadır.

İHA için rota maliyeti aşağıdaki gibidir.

Bir rotanın maliyeti hesaplanırken, önce rotanın mesafesi hesaplanmakta daha sonra rota üzerinde herhangi bir engel olup olmadığı tespit edilip, var ise amaç fonksiyon değerine ek olarak engel maliyeti de eklenmektedir. Uzaklık hesaplarken dünyanın geometrik şekli göz önüne alınmıştır. Geometrik şekli dikkate alacak şekilde Haversine (Montavont ve Noel 2006) formülü kullanılmıştır. Haversine formülü enlemi ve boylamı bilinen iki koordinat arasındaki mesafenin hesaplanması için kullanılmaktadır.

D_{enlem} = Enlemler arasındaki fark

D_{boylam} = Boylamlar arasındaki fark

R = 6,371km (dünyanın yarıçap)

d = İki koordinat noktası arasındaki mesafe

h = haversin (D_{enlem}) + cos(enlem1) * cos(enlem2) * haversin(D_{boylam})

haversin (S) = $\sin^2$ (S / 2)

d = 2 * R * arcsin (sqrt h)

Öncelikle herhangi bir engele çarpmadan veya engelli bir alana girmeden rota planlaması yapılması gerektiğinden, engelli alanların belirlenmesi gerekmektedir. Rota üzerindeki engellerin tespiti için, denklemi bilinen bir doğrunun bir noktaya olan uzaklığı formülü kullanılmıştır. A(x_1 ,y_1) ve B(x_2 , y_2) koordinatlara sahip iki nokta arasında herhangi bir engel var mı yok mu doğrunun denklemi ile bulunur.

$$\frac{y_2 - y_1}{x_2 - x_1} = \frac{y - y_2}{x - x_2}$$

Denklem sonucunda a = ($y_2 - y_1$) , b = ($x_2 - x_1$) ve c = (y_1 * $x_2 - y_2$ * x_1) olarak bulunmaktadır. O (x_3 ,y_3) merkezli bir engel ile doğru arasındaki uzaklık aşağıdaki denklem ile hesaplanır.

$$d_{engel} = \frac{|ax_3 + by_3 + c|}{\sqrt{a^2 + b^2}}$$

d_{engel} → Engel yarıçapına eşit veya küçük olursa yolumuz da engel var demektir. Eğer büyükse oluşturulan rotada engel yok demektir.

Rota üzerindeki noktalar arasında bulunan engel sayısı kadar, mevcut rotanın amaç fonksiyon değerine engel maliyeti eklenmektedir. Rotanın amaç fonksiyon değerine engel maliyeti eklenmektedir. Bu bilgiler ile rotanın amaç fonksiyon değeri

$f_i = d_i + E_i$

f_i = i. rotanın maliyetini

d_i = i. rotanın uzunluğunu

E_i = i. rota üzerindeki toplam engel sayısı ve engel maliyetinin çarpımını gösterir.

denklemi ile hesaplanır.

Takip edilecek rota üzerinde bulunan coğrafi engeller (dağ, tepe vb.) de dikkate alınmaktadır. Engellerin tespit edilmesi için tarama mesafesi kullanılmıştır. Yüksekliklerin belirlemesi için başlangıç noktasından hedef noktasına kadar olan rota üzerinde herhangi bir coğrafi engel olup olmadığının kontrolü için yükseklik matrisi oluşturulmuştur. Tarama mesafesi, rota uzunluğuna göre orantısal olarak değişmekle birlikte, sabit olarak da kullanılmıştır. Tarama mesafesi 50m olarak alındığında rota boyunca her 50m'de bir coğrafi engel olup olmadığı ve

varsa yüksekliği kontrol edilmektedir. Rota üzerinde coğrafi engelli bölgeye yaklaşıldığında, bu engellerden sakınım için, İHA'nın kendi etrafında dönerek engelleri aşması hedeflenmiştir. İHA'nın kendi etrafında dönerek aşması için ulaşması gereken yükseklik aşağıdaki şekilde hesaplanır.

h = 2πr * tan (a)

h = İHA'nın kendi etrafında bir tur döndüğünde aldığı yükseklik mesafesi.

a = Yükseliş açısı

r = İHA'nın kendi etrafında döneceği minimum yarıçap değeri

h değeriyle İHA kendi etrafında kaç tur dönecek onu öğreneceğiz.

Bu bilgiler ile yapay arı kolonisi faydaları aşağıdaki gibidir.

- Uygulanması kolay.
- Komşu yerlerde farklı çözüm arayışları var.
- Doğadan ilham ile daha basit ve sağlam.
- Birçok farklı alanda kullanılabilir.

Algoritmanın en temel hali

Döngü {

İşçi arılar

Gözcü arılar

Kaşif arılar

En iyi çözümü tut

} Çevrim maksimum döngü sayısında yada tanımlanan kriter olanan kadar

Bu iki algoritmayı ayrı ayrı kullanabileceğimiz gibi birleştirip karışık çözüm yapabiliriz.

3. YAPAY ARI KOLONİSİ VE RASTGELE AĞAÇLAR YÖTEMİ İLE ÇÖZÜM

Çözümün başı rastgele ağaçlar algoritması ile başlayıp sonu yapay arı kolonisi algoritması ile devam etmektedir.

Haritayı çağır

Başlangıç ve bitiş noktalarını belirle

while (i<düğüm sayısı veya hedefe ulaşana kadar)

{ Harita üzerinden rastgele q_{rand} düğümlerini belirle

for (Başlangıç noktasına en yakın q_{near} seç)

{ Başlangıç noktasından q_{near} düğümüne dal oluştur,

q_{near} düğümünden q_{rand} düğümü yönüne yeni

dallanmalar için yönlendir

if (Düğüm çok uzaksa)

{ Enterpolasyonla düğümü yakınlaştır q_{new} düğümüne ulaş

Engele çarpmadığını kontrol et,

q_{near} düğümü ile q_{new} düğümü arası

C_{min} maliyet fonksiyonunu hesapla,

Düğüm listesine q_{new} düğümünü ekle }

end if }

end for }

end while }

Belirlenen minimum yolu belirleyen tüm düğümlerinin x ve y değerlerini hafızaya at

Burada rastgele ağaçlar algoritmasının yardımı ile yapılan daldaki düğümler

$path_{x,y} = \sum_{i=2}^{n-1} N_i$ { i = 1,2,…,n }

n = yolu çizilen düğüm sayısı.

Formülü ile hafızaya atılıp yapay arı kolonisinin kullanımı için bekler. Yapay arı kolonisi algoritması bundan sonra devreye girmektedir.

Repeat:

for (Engel yoksa ve k<M iken)

{ Kaynak oluştur $F_{x,y} = \sum f_{x,y}$ M *{ k=1,2,3…M } ,*

M kaynak sayısıdır}

end for

for (i<n)

{ RRT'de belirlenen düğüm noktalarını (ABC'de kaynak)

$N_i = N_i + \Phi_i (N_i - F_{k,j})$ formülü ile yenileri ile değiştir,

Yeni düğümler için $obj = W_p * Pl + W_a Sm * + W_s Sa$ (maliyet

fonksiyonu) fonksiyonunu hesapla

if (Yeni belirlenen düğümün maliyet fonksiyonu eskisinden daha iyi ise)

{ Hafızadaki eskinin yerini yeni düğüm ile değiştir}

end if

Düğümlerin olasılıksal değerini hesapla

Olasılık değerine göre rastgele düğüm oluştur

En iyi olan yolu hafızaya kayıt et }

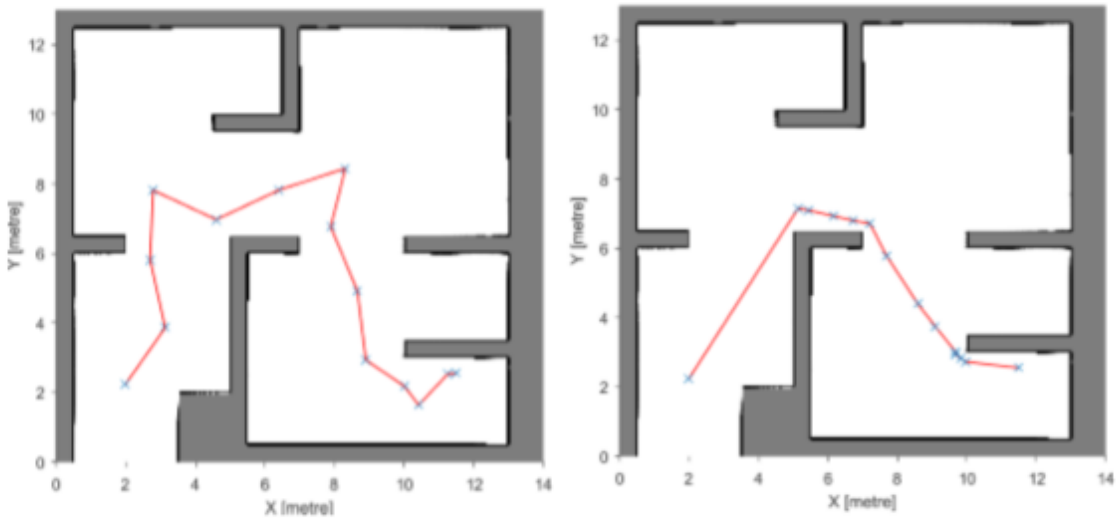
end for

until (İterasyon sayısına ulaşana kadar tekrar et)

	RRT	ABC**	RRT-ABC
Yol Uzunluğu (Pl), metre cinsinden	18.3386	19.1253	14.4993
Yumuşaklık-Akıcılık Değeri (Sm)	37.3402	52.7178	18.5757
Güvenlik Değeri (Sa)	0.3183	0.4818	0.3616
Maliyet fonksiyonu (obj fonksiyonu)	38.412	66.3731	30.1372
Bilgi işlem zamanı (saniye)	6.54	14.162	11.17

Tablo : Rastgele ağaçlar algoritması ve yapay arı algoritması karşılaştırılması

RRT ve RRT-ABC algoritmalarının haritada bulduğu yollar Şekil 1.a ve Şekil 1.b'de sunulmuştur.



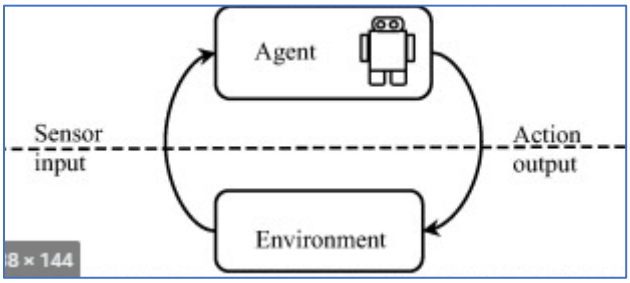
(a) (b)

Şekil 1. a. *Başlangıç (B) ve sonuç (S) noktası belli haritada çalıştırılmış RRT*
b. *Başlangıç (B) ve sonuç (S) noktası belli haritada çalıştırılmış melez RRT-ABC*

4. ZEKİ AJANLAR

Çevreden bilgi alıyor. Aldığı bilgiler üzerine faaliyetler gerçekleştiriyor.

- Biyolojik ajanlar = Hayvanlar, organizmalar.
- Software ajanlar = Virüs, sohbet robotları.
- Hardware ajanlar



Evlerimizdeki normal kapıların açılıp kapanması kapı kolu ile yani bizim yardımımızla olurken alışveriş merkezleri, havaalanı gibi yerlerde kullanılan hareketli kapılar sensörler ile kapıdan geçecek biri olduğunu anlayıp otomatik olarak açılıyor. Otomatik kapı sistemlerine bir tür ajan diyebiliriz.

F: p*-> A

P= girdiler duyargalar

A= çıktılar aksiyonlar

Rasyonel Ajanlar = Rasyonel ajandan beklenen otomatik pilot ile uçan uçağın çıkış noktasından varış noktasına az yakıt harcayarak güvenli bir şekilde gitmesi iken rasyonel olmayan otomatik pilottan şekil çizerek, yazı yazarak ilerlemesi beklenir.

Rasyonel ajan oluştururken amacımız en kısa zamanda minimum enerji ile işlerini yapabilmesidir.

Otonom sistemlerde girdilere göre kendi kendine karar vererek eylemleri yapması beklenir.

Ajanlar için kriterler;

- Performans = Güvenlik, maliyet, hız…
- Aksiyonlarımız = Hızlanma, durma…
- Çevre = Yollar, trafik…
- Sensörler = Kamera, sonar, hız göstergesi…

Otomatik drone için örnek;

- Performans: Hızlı, az enerjili çalışması, ani manevralar yapması
- Çevre: Hava, uçan diğer canlılar, çevredeki diğer dronelar, ağaçlar, binalar ve diğer engeller
- Aksiyon: Sağa-sola dönme, yükselme, alçalma
- Sensörler: Kamera, hız göstergesi, GPS, altimetre, hareket sensörü, ultrasonik sensör

Ortam çeşitleri

- Tamamen bilinen ya da bir kısmı sadece görerek tanımaya çalışılan ortam
- Deterministik bir sonraki adım durumunun biliniyor olmasıdır. Olasılık içeren olasılıksak bir yapı ise stochastic bir yapıdır.
- Episodik: Asıl işin başlayıp bitmesidir. Problem küçük alt parçalara bölünüyorsa episodik değildir.
- Statik: Ortam değişmiyor ise statik, değişiyor ise dinamik.
- Ayrık ortamın kesikli olması, sürekli ise ortamın devamlı olmasıdır. Bir zamandan diğer zamana geçerken eylemlerin sürekli olmasıdır. Ayrık, ara durumlar olmadan durumdan duruma geçiş yapmasıdır, ara durumları ile alt yapıları devamlı değişiyor ise sürekli olmasıdır.

Tek ajan= Tek başına çalışıyor eğer birbirleri ile ortak iş yapıyorlarsa çok ajanlı sistemler deriz.

Ajan fonksiyon ve programları

Aldığı verilere göre hemen faaliyete geçer, aldığı verileri analiz edip başka durumları da düşündükten sonra faaliyette bulunur.

Ajan çeşitleri;

-Basit refleks ajanları

-Model tabanlı refleks ajanları

-Hedef tabanlı ajanlar

-Fayda tabanlı ajanlar

Basit Refleks Ajanlar: Sensörlerden giren bilgileri kurallara göre değerlendirip faaliyete geçer. Porttan gelen bir paket engeller listesindeyse engeller, engeller listesinde değilse engellemez.

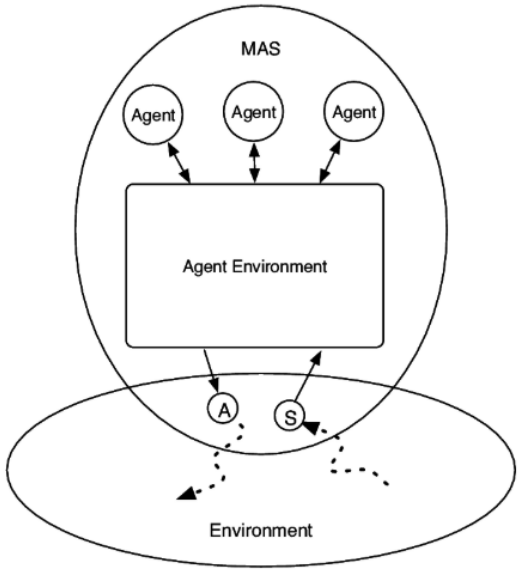
Model Tabanlı Refleks Ajanlar: Sensörden gelen girdilere göre durum analizi yapılıyor. Durumlar arası analiz yapılarak aksiyon seçimi sonucunda neler olacağını hangi duruma geçeceğimizi belirliyor. Buna göre durum seçimi ve aksiyonumuzu yapıyoruz. Bu tür ortamı daha iyi analiz eder.

Hedef Tabanlı Ajanlar: Sensörlerden gelen bilgi ile ortam nasıl etkileniyor ve aksiyon seçimine göre hedefe ne kadar yaklaşacağımızı belirliyor.

Fayda Tabanlı Ajanlar: Sensörler ile bulunduğumuz durumu ölçüyoruz. Ortam değişimi ve yapacağımız faaliyetlere göre hedefe ulaşma aşamasında nasıl daha faydalı ilerleme yapabiliriz.

ÇOK AJANLI

Çok ajanlı sistemleri kontrolünün sorunları büyük bir öneme sahiptir. Her çok ajanlı robot sistemi, birkaç mobil robottan oluşan bazı taşıma alt sistemine sahiptir.



Bu tür mobil robot grubunu kontrol etme problemi iki ana bölüme ayrılabilir:

– Optimal global (genel) görev alt görevlere ayrıştırılması ve gruptaki ayrı robotlar arasındaki optimum dağılımı.

- Her mobil robot için yol planlama, kontrol ve hareket düzeltme.

Bir çalışma bölgesindeki mobil robot-ajanların yol planlaması ve kontrolü sorununu iki alt probleme ayırmak mantıklıdır.

- Ajan diğer ajanların hareketini hesaba katarak her bir ajan için ayrı ayrı yol planlama yapar. Bu sorun, engelsiz bir ortamda yol planlama ve optimizasyonun düzeltme algoritmaları ile çözülebilir. Bu aşamada, çevre hakkında tam bilgili olduğu varsayılır (ortam bilinmeyen engeller içermiyorsa).

- Çevre hakkında bilgi eksik olabilir veya ajan planlanan yollardan sapabilir. Bu sorunu çözmek için iki temel alternatif vardır. Birincisi, yolları çeşitli yol lokal düzeltme algoritmaları vasıtasıyla düzeltmektir. Bu yaklaşımın eksikliği optimal olmayan ajan duygularıdır. İkinci yöntem, keşfedilen yeni engeller veya çarpışmalar meydana geldiğinde tam veya yerel olarak optimum yol yeniden planlamasıdır.

Her çevre modelinin, yol planlama amaçları için belirli avantajları ve dezavantajları vardır. Örneğin:

Izgara modelinin kullanımı kolaydır, farklı ajanlar tarafından toplanan verilerle düzeltilir ve güncellenir. Ancak yüksek bellek masrafları gerektirir, ayrıca yüksek veri yedekleme ve doğruluk eksikliği vardır. Bu dezavantajlardan bazıları daha kapsamlı ızgara modeli kullanılarak ortadan kaldırılabilir.

Vektör modelleri yüksek hassasiyet, düşük bellek masrafları içerir, ancak bu tür modelleri kullanarak bir yol planlamak zordur, aynı zamanda çevreyi güncellemek de zordur sensör

bilgilerinin genellikle ayrık biçimde sunulduğu ve bu nedenle vektör formuna dönüştürülmesi gerektiğinden, robotların sensör sistemlerinden gelen verilerle yapılmaktadır.

Grafik model, yol ve hareket planlama problemleri için daha uygundur. Kural olarak, grafik modeli sadece olası yollardan oluşur, yani grafik oluşturma sırasında engellerle ilgili bilgiler hariç tutulur. Izgara ve Vektör modelleri grafik modeline eşlenebilir. Grafikte optimizasyon problemini çözmek için bilinen çeşitli algoritmalar vardır, örneğin Dijkstra algoritması. Vektör ortam modelindeki olası yollar bir görünürlük grafiği ile temsil edilebilir. Görünürlük grafiği, düğümlerin çokgen engellerin köşelerini temsil ettiği bir grafiktir ve kenarları, engel köşelerini, yani "görünürlük" çizgilerini bağlayan düz olası yolları temsil eder. Statik grafik oluşturulduktan sonra, hedef ve başlangıç noktaları eklenir ve diğer grafik düğümlerine bağlanan görünür kenarlar hesaplanır. Yolları planlamak için grafik modeli daha fazla kullanılacaktır. Olası yolların grafiği hem vektör hem de ızgara ortamı modellerinden elde edilebilir. Ayrıca, kabul edilebilir yolların grafiği, ajanların hareket deneyimi temelinde inşa edilebilir. Ajanların hareketleri hakkındaki bilgiler ayrı olarak veya grafik düğümlerinde saklanabilir.

Özetle, hareket planlaması için bir görünürlük grafiği veya olası yolların grafiğini kullanmanın avantajları aslında 2D veya 3D yapılandırma aralığında optimal yollar veren basit, iyi anlaşılmış bir yöntem olmasıdır.

Çok aracılı optimal yol planlaması için kullanılan grafik ortamı modeli aşağıda açıklanmıştır. Ortamdaki noktalar (yerler) ve aralarındaki kabul edilebilir (olası) yollar, düğümleri ortamdaki belirli yerleri ve kenarları kabul edilebilir yolları temsil eden grafikle temsil edilir. Grafiğin her bir kenarının, ilgili düğümler arasındaki yol uzunluğu, seyahat süresi veya hareket zorluğu, vb. İçin yeterli bir ağırlığı vardır. Grafik oluşturma işlemi yalnızca kabul edilebilir yollardan oluşur.

Bir grafik G<V,E>, M tane düğüm düşünelim. Tüm düğümler numaralandırılmıştır. Her düğüm, $M_i \geq 1$ bitişik düğümlerine (köşeler) i_1 , i_2 , ..., i_{Mi} 'ye sahiptir. Ayrıca, tüm grafik düğümleri bir ağırlık W_i bir düğüm i (i = 1,2, ..., M) ile en aza indirilmiş işlevselliğin (örneğin, mesafe veya hareket süresi) değerine karşılık gelir. Grafiğin her bir kenarına, düğümleri i ve j'yi bağlamak için iki özellik atanır:

S_{ij} - bu iki düğüm arasındaki boşluk mesafesi ve hareket hızına bağlı olarak hareket süresi. Özetle, bu tür herhangi bir grafik aşağıdaki gibi özelliklere sahiptir:

Her bir Grafik Düğümü,

-Çevre uzayındaki bir noktanın koordinatları.

- W_i değeri minimize edilecek işlevsel (mesafe, zaman, vb.).

- Bitişik düğüm kümesi i_1 , i_2 , ..., i_{Mi} .

-Ajanlar düğüm boyunca hareket etmesi gibi çok ajanlı yol planlaması için gerekli ek özellikler vardır.

Grafiğin her kenarı aşağıdakilerle karakterize edilir:

- S_{ij} düğümü arasındaki mesafe i ve j arasındaki mesafedir.

- Kenar l_{ij} ağırlığı, düğüm i'den j'ye hareket zamanına karşılık gelir. Bu değer değişkendir ve yol planlanırken değiştirilebilir.

- Ek özellikler. Örneğin, kenar, i ve j arasındaki hareket yönüne bağlı olan iki farklı ağırlık l_{ij} ve l_{ji} 'ye sahip olabilir. 3 boyutlu ortamı veya iki yönlü yolları simüle etmek için izin verir.

Yol planlamadan önce, grafik kenarlarının tüm ağırlıkları aşağıdaki gibi başlatılmalıdır:

$$l_{ji} = S_{ji} / V$$

V = Ajanın ortalama hızıdır.

Başlangıç düğümü t = 0 anında başlatılmalıdır. Yol planlaması sırasında, ajanın bu düğümlerden geçtiği zaman düğümlerin ağırlıkları değişir ve zamanın anlarına eşit olur.

Ajanlar yollarını sırayla planlar ve bir sonraki robot yolunu planlarken, çarpışmayı ortadan kaldırmak için önceden planlanmış tüm yollar dikkate alınır.

Tek ajandan geliştirilen çok ajanlı optimal yol planlama algoritmasının ana farkları aşağıdaki gibidir:

- Her grafik düğümünde i (i = 1,2, ..., M) sadece kendi ağırlığı W i değil, düğüm ayrıca iki dizi saklar: diğer ajanlar bu düğüm i (t_{ji} ajan j'nin i düğümünden geçtiği bir zamandır) düğümünden geçtiği bir zamandır ve bu ajanların robotlarının kimliklerini de içerir.

- Çarpışmalardan kaçınmak için her robotun bir yolunu planlarken grafik (özellikle kenarların ağırlıkları) değiştirilebilir.

Çok ajanlı yol planlaması çarpışmasız planlama sağlayan bir dizi kural ile yapılmalıdır. Bununla birlikte ajanın biri hızlanabilir biri yavaşlayabilir. Eğer D * algoritması temel bir yol optimizasyonu algoritması olarak kullanılıyorsa, iki düğüm arasındaki mesafe değiştirilebilir. Mesafenin değiştirilmesi, ortam modeli düzeltmesine karşılık gelir.

Grafik düğümü ağırlıklarının başlatılması W_i (i = 1,2, ..., M), tek ajanlı yol planlama algoritması ile aynıdır ve her robot yolu planlamasından önce gerçekleştirilir. Herhangi bir robot için bir yol planlarken, grafik düğümü ağırlıkları tek aracılı yol planlama algoritmasında olduğu gibi değiştirilir:

$$W_i = \begin{cases} W_i + I_{iij}, \text{ if } (W_i + I_{iij}) < W_{ij}, \\ W_i, \text{ if } (W_i + I_{iij}) \geq W_{ij}, \end{cases}$$

Yol düğümlerine karşılık gelen grafik düğümlerinde çarpışmaları önleyici kurallar ve düz yollara karşılık gelen grafik kenarlarında çarpışmalardan kaçınmak için kurallar yapılmıştır.

$W_i + I_{iij} = t_{kij}$ (k = 1,2, ..., n) ise grafik düğümlerinde çarpışmalardan kaçınmak, burada n bir grup ajan ise, o zaman

$$I_{iij} = I_{iij} + e$$

e = aynı düğümden geçen farklı robotlar arasındaki minimum zaman aralığı.

e güvenli düğüm geçişi sağlamalıdır. ij düğümünün ağırlığı W_{ij} formüle göre hesaplanır. Hız, yolda parça halinde sabittir ve her kenar için i ve j düğümlerini şu şekilde bağlanarak hesaplanır:

$$V_{ij} = S_{ij} / I_{ij}$$

$t_i < t_{kij}$ ve $W_i + I_{iij} = t_{kij}$ (k = 1,2, ..., n)olduğunda grafik kenarlarında çarpışmalardan kaçınma denklemimiz, $t_{ki} > t_{kij}$ ise iki ajanın zıt yönlerde hareket edeceği ve planlanan ajanın ajan k'den önce kenardan geçeceği bir durumdur. Çarpışma olmayacağı için kenar ağırlığının değişmesine gerek yoktur. Bir sonraki düğüm W_{ij} 'in ağırlığı şu şekilde hesaplanır:

$$W_{ij} = \begin{cases} W_i + I_{iij}\text{ , if } (W_i + I_{iij}) < W_{ij}\text{ ,} \\ W_i\text{ , if } (W_i + I_{iij}) \geq W_{ij}\text{ ,} \end{cases}$$

$$(t_i < t_{kij}) \wedge [t_{kij} \leq [W_i (t_{kij} - t_{ki}) + I_{iij} * t_{ki}]/ I_{iij} = t_{kij} - t_{ki} - I_{iij}] \leq t_{kij}]$$

Eğer yukarıdaki gibi bir durum varsa çarpışma mümkündür: ajan k, planlanmakta olan ajanı takip edecek ve kenara çarpacaktır. Çarpışmayı önlemek için mevcut ajanın hızı artırarak hareket süresini (kenar ağırlığı) değiştirmek gerekir.

$$I_{iij} = (W_i - t_{kij} - e) (t_{kij} - t_{ki}) /(t_{ki} - t_{kij} - e)$$

Ve tekrar tekrar W_{ij} düğüm ağırlığı hesaplanmalıdır.

$$W_{ij} = \begin{cases} W_i + I_{iij}\text{ , if } (W_i + I_{iij}) < W_{ij}\text{ ,} \\ W_i\text{ , if } (W_i + I_{iij}) \geq W_{ij}\text{ ,} \end{cases}$$

Ajan k hangi yolun planlanmakta olduğunu takip eder. Daha sonra kenar ağırlığı değiştirilmeyecek ve düğüm ağırlığı W_{ij} şu şekilde hesaplanır:

$$W_{ij} = \begin{cases} W_i + I_{iij}\text{ , if } (W_i + I_{iij}) < W_{ij}\text{ ,} \\ W_i\text{ , if } (W_i + I_{iij}) \geq W_{ij}\text{ ,} \end{cases}$$

$$(W_i > t_{ki}) \wedge [(W_i + l_{iij}) > t_{kij}] , (k=1,2,...,n) \text{ birde bu varsa } t_{ki} > t_{kij}$$

Ajanlar zıt yönlerde hareket ettiğinde, ajan k kenarı diğer ajandan daha erken geçecekse kenar ağırlığını değiştirmeye gerek yoktur. Sonraki düğüm ağırlığı W_{ij} şu şekilde hesaplanır:

$$W_{ij} = \begin{cases} W_i + I_{iij}\ , \text{if } (W_i + I_{iij}) < W_{ij}\ , \\ W_i\ , \text{if } (W_i + I_{iij}) \geq W_{ij} \end{cases}$$

Eğer aşağıdaki durum olursa,

$(t_{ki} < t_{kij}) \wedge [tki \leq [W_i (t_{kij} - t_{ki}) + I_{iij} * t_{ki}]/ I_{iij} = t_{kij} - t_{ki} - I_{iij}] \leq t_{kij}]$

Yol planlayan ajan, ajan k 'yı takip edecek ve yüksak hız nedeni ile ajan k'ya çarpacak. Çarpışmayı önlemek için hızını azaltmak ve kenardaki hareket süresini artırmak için

$I_{iij} = (W_i - t_{kij} - e) (t_{kij} - t_{ki}) /(t_{ki} - t_{kij} - e)$

Formülü kullanılır. Düğüm ağırlığı tekrar hesaplanır

Eğer $(W_i < t_{ki}) \wedge [(W_i + l_{iij}) > t_{kij}]$, (k=1,2,...,n) olursa

$t_{ki} < t_{kij}$ durumunda çarpışma mümkündür. Ajan k düğüme çıkmadan önce planlanmakta olan rotayı takip edecek ve vuracaktır. Çarpışmayı önlemek için ajanın hızı arttırılmalıdır. Kenar ağırlığı formüle göre değiştirilecektir.

$I_{iij} = (W_i - t_{kij} - e) (t_{kij} - t_{ki}) /(t_{ki} - t_{kij} - e)$

Daha sonrasında tekrar düğüm ağırlığı hesaplanır.

Eğer $t_{ki} > t_{kij}$ ise çarpışma kesin olacak. Ajan k kenarda ters yönde hareket ediyor olacaktır, yol planlaması yapan ajan kenara doğru ilerler. Mevcut ajan için i düğümünden ij düğümüne hareket yasaklanırsa çarpışma önlenmiş olur. Hedefe ulaşmak için kenar ağırlığını değiştiriyoruz.

$l_{iij} = \infty$

En sonunda tekrar W_{ij} 'nin düğüm ağırlığı hesaplanır. En son oluşturulan kenar sonsuz ağırlığı nedeni ile yola dahil edilmez.

Eğer $(W_i > t_{ki}) \wedge [(W_i + l_{iij}) < t_{kij}]$, (k=1,2,...,n) olursa

$t_{ki} < t_{kij}$ çarpışma olacak tek olaydır. Çarpışmayı önlemek için kenar ağırlığını

$I_{iij} = (W_i - t_{kij} - e) (t_{kij} - t_{ki}) /(t_{ki} - t_{kij} - e)$

Formüle göre değiştirmek gerekir. Ve tekrar düğüm ağırlığı hesaplanır. Algoritma sonlanır.

5. A* ALGORİTMASI

Eşlenmiş bir alanda iki konum arasındaki en kısa yolu bulmak için oluşturulan algoritmadır.

Algoritma daha önceden öğrenilen alanları saklar. Bu saklama işlemi için kapalı liste oluşturur ve yanındaki alanları saçak listeye kayıt eder.başlangıç ve bitiş noktaları verilir. Başlangıçtan bitiş noktasına gelene kadar kapalı ve saçak listeleri büyür.

Algoritmada mesafe değerlendirmesi:

f (n) = g (n) + h (n)

g (n) = Başlangıçla herhangi bir köşe arası maliyet

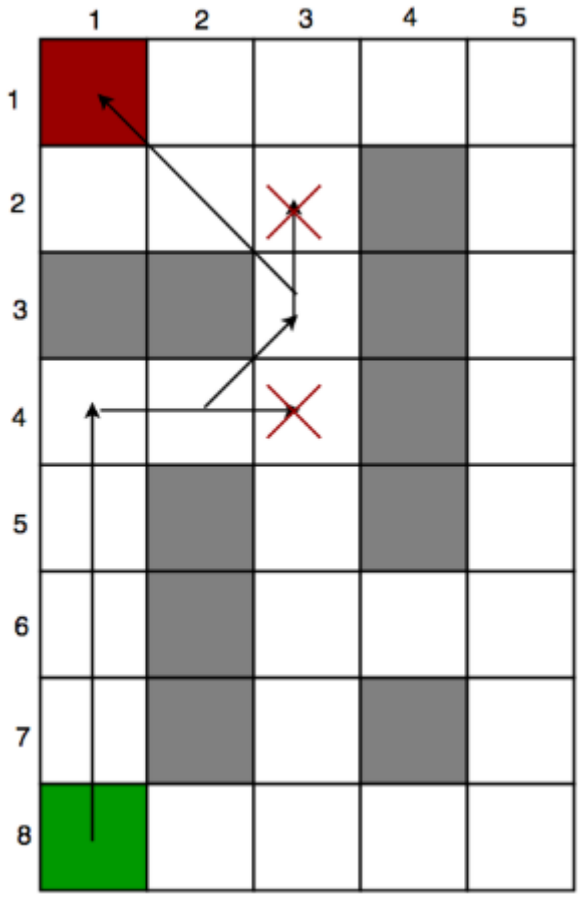
h (n) = maliyet hesabının sezgisel olarak yapılması (kuş uçuşu mesafe)

h(n) hesaplama :

x2 = hedef konumun koordinatı
x1 = geçerli konumun koordinatı
y2 = hedef konumun koordinatı
y1 = mevcut konumun koordinatı
dx = | x2 – x1 |
dy = | y2 – y1 |

$h(n) = \text{sqrt}(dx^2 + dy^2)$

Her nokta seçiminde algoritma başa sarar ve hedefe ulaşana kadar devam.



Algoritma her zaman doğru sonuçlar vermeyebilir. Doğru sonuç olup oladığını aşağıdakiler ile test edebilir.

-H (n) = gerçek maliyet

Uygun sonuç.

-H (n) < gerçek maliyet

Gidilen yoldan zarar ederiz.

-H (n) > gerçek maliyet

En uygun sonuç. Karlı bir iş.

6. GRAF TEORİSİ

Basit mantığı elimizi kaldırmadan şekil çizmeyi amaçlayan yöntemdir. Grafın temelinde matematiksel bir nesnedir. Graflar düğümler ve kenarlardan oluşur. Graflar sadece yol ile ilgili değil insanlar arasındaki ilişkiyi de gösterebilir. Organ bağışı sırasında da kullanılabilir. Kısaca birden fazla varlık ve aralarında ki ilişki söz konusu olduğunda grafa başvurabiliriz. Düğümler varlıkları kenarlar ilişkiyi gösterir. Yönlü, yönsüz, ağırlıklı, döngülü graflar vardır.

Yönlü Graf = Bağlantı üzerinde hangi düğümden hangi düğüme geçebiliriz onu gösteriyor.

Ağırlıklı Graf = Kenar ağırlıkları olan graflardır. (uzaklık, maliyet, zaman gibi kavramları belirtir.)

Döngülü Graf = Kenarın bir düğümden çıkıp yine aynı düğüme gelmesi ile oluşur.

Derece = Bir düğümden çıkan kenar sayısı o düğümün derecesidir.

G = (V, E)

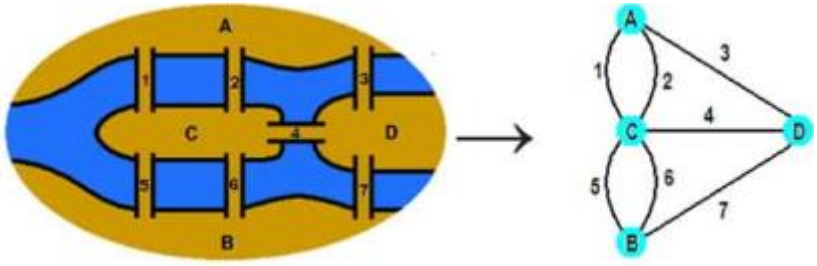
V = Düğüm

E = Kenar

Ağaçlar, döngü olmayan graflardır. İki düğümü bağlayan tek bir kenar vardır. Düzenli graflarda tüm dğümlerin dereceleri birbirine eşittir.

6.a. EULER YOLU

Kenarların üzerinden bir kere geçilerek her kenardan bir kere geçilebilir mi? Bu sorun ilk olarak Königsberg Köprüsü ile ortaya çıkmıştır.



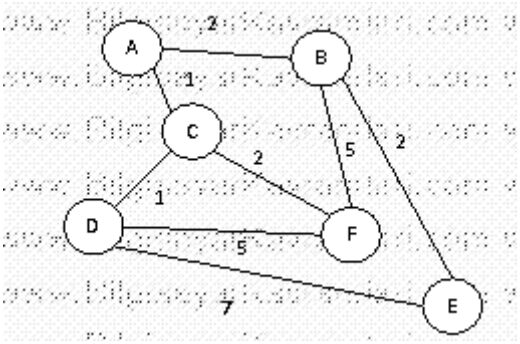
Problem bütün köprülerden bir kere geçilen bir yol olup olmayacağıdır. Buna göre şayet bir düğümün derecesi tekse, bu düğüm ya başlangıç ya da bitiş düğümü olmalıdır. Bunun dışındaki durumlarda (yol üzerindeki herhangi bir düğüm olması durumunda) tek sayıdaki yolun sonuncusu ziyaret edilmiş olamaz. Düğümlerin dereceleri 3 ve 5 sayları olduğu en başta sorulan sorunun cevabı hayırdır.

Euler yolu tanımlamasına gelirsek yönsüz bir grafta bütün düğüleri dolaşan bir yol varsa bu yola euler yolu diyoruz.

Euler yolu olabilmesi için düğüm derecelerinin her biri çift sayıda olmalı yada sadece iki tane tek sayıda düğüm içermeli diğer kalan düğümler çift sayıda olmalı. Tek sayıda olan düğümler başlangıç ve bitiş noktası olarak ayarlanması gerekiyor.

6.b. DİJKSTRA ALGORİTMASI

Algoritma başlangıç düğümünün çevresindeki kenarları inceler ve en düşük maliyetli kenarı seçerek yeni düğüme ilerler. Bu şekilde en ucuz, kısa yolunu bulmayı amaçlar.



A düğümünü başlangıç olarak alırsak c'ye ve b'ye bağlı iki kenarı var. Bu durumda küçük olan yolu seçiyoruz, 1 ve c düğümüne ilerliyoruz. C'nin komşuları olan d ve f'den daha uygun olan maliyeti olan 1'i seçip d düğümüne ilerliyoruz. D'nin komşuları olan kenarlar 5 ve 7 den küçük olan 5'i seçip f düğümüne geçiyoruz. Daha önce bulunduğumuz düğümlere tekrar uğramıyoruz. F düğümünde iken komşu olan düğümlerden birine uğrandığı için uğranan düğümü yok sayıyoruz. Gidebileceğimiz tek komşu olan b düğümüne gidiyoruz. Tek düğüm hariç hepsine uğrandığı için sona kalan düğüm e de yolumuzu bitiriyoruz.

```
For all nodes N of graph G(N,E)
        If N is not Start
                Cost(N)= ∞ and Insert (NotVisitedNodeList,N)
        Else
                Cost(N)=0 and Insert(VistedNodeList,N)
While NotVisitedNodeList≠ Ø
Find N with Min Cost(N) in NotVisitedNodeList
                For all edges E<N,X> of Node N
                        If Cost(X)>Cost(N)+Weight(E)
                                Cost(X)= Cost(N)+Weight(E)
                                X.Predecessor=N
                        Insert(VistedNodeList,N)
                        Delete (NotVisitedNodeList,N)
```

Algoritmanın yalancı kodu yukarıdaki gibidir.

KAYNAKLAR

- Yunis TORUN, Züleyha ERGÜL, Ahmet AKSÖZ, Optimum Enerji Verimliliğini Hedefleyen Rastgele Ağaçlar ve Yapay Arı Kolonisi Yöntemi ile Otonom Robotlarda Yol Planlama Algoritması, 2019.
- Çavuş, Tuncer / İnsansız Hava Araçları İçin Yapay Arı Kolonisi Algoritması Kullanarak Rota Planlama, 2017.
- Erhan BÜLBÜL, DEĞİŞEN 3B ORTAMDA EN KISA YOL ALGORİTMALAR, 2015.
- Hülya YILMAZ, RANDOM FORESTS YÖNTEMİNDE KAYIP VERİ PROBLEMİNİN İNCELENMESİ VE SAĞLIK ALANINDA BİR UYGULAMA
- Youtube video linkleri:

https://www.youtube.com/watch?v=paMcKZlcv78

https://www.youtube.com/watch?v=7daxVRoKwIA

- Kullandığım internet siteleri

http://bilgisayarkavramlari.sadievrenseker.com/2010/05/13/dijkstra-algoritmasi-2/

- Bunların yanında bilgi aldığım konuyu kavramama yardımcı olan Youtube videoları ve diğer ufak sitelerde var.