

**İstanbul Teknik Üniversitesi Robotik Arama Kurtarma Ekibi**

**Görev Raporu**

**Görev Konusu:** Robotlarda Yol Bulma Algoritmaları ve Robotikte Sürü Tipleri

**Hazırlayan:** Önder İhsan Kul

**Teslim Tarihi:** 05.10.2023

**Robotlarda Kullanılan En Kısa Yol Bulma ve Yol Planlama Algoritmaları**

**En Kısa Yol Algoritmaları**

**Best First Search**

Best First Search “Ağaç” yapısındaki graph’larda verilen bir kaynak ve hedef arasındaki en kısa yolu hep en kısa kenarları kullanarak tahmin etmeye çalışan algoritmadır. Hızlıdır ancak her zaman doğru veri sağlayacağı kesin değildir.

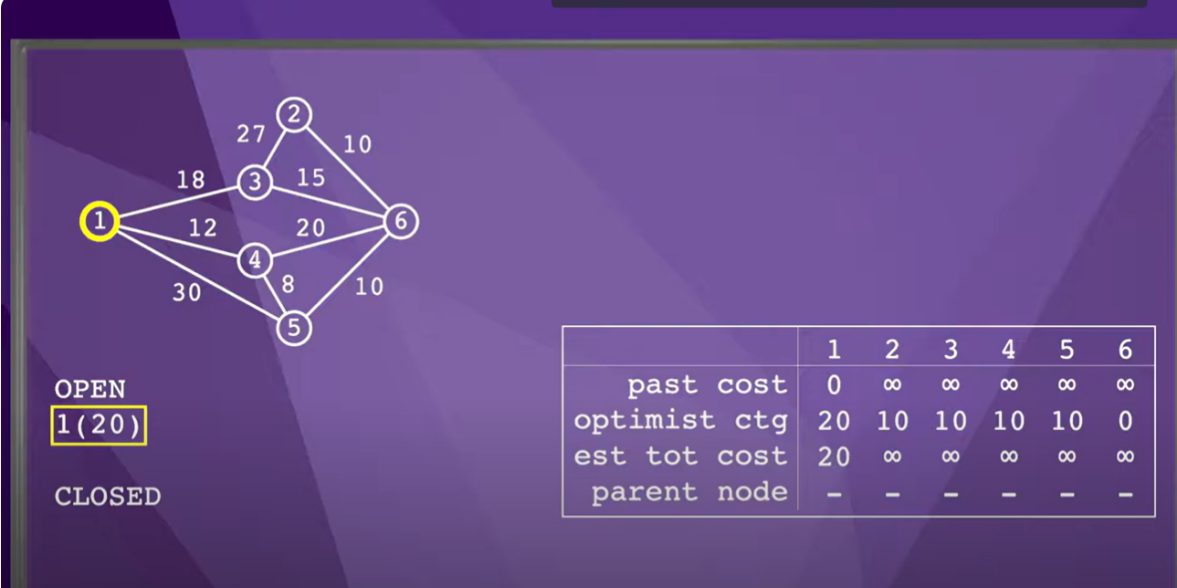
**Dijkstra En Kısa Yol Algoritması**

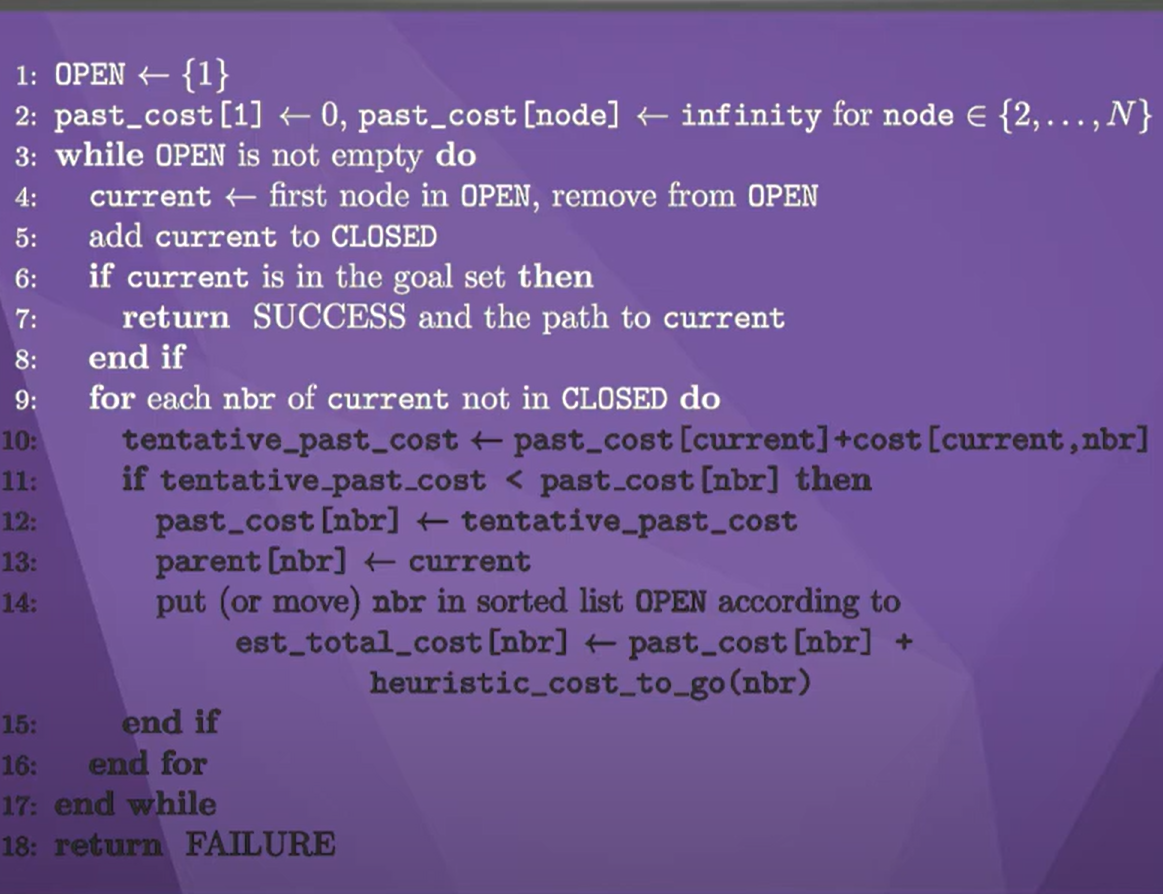
Dijkstra en kısa yol algoritması, Hollandalı bilgisayar bilimci Dijkstra tarafından oluşturulmuştur. Negatif değerli kenar içermeyen yönlü ve yönsüz graph’larda kaynak bir düğümden diğer tüm düğümlere olan en kısa yolu bulmaya yarar. Kesin sonuç vermesi aşağıda bahsedeceğim bazı algoritmalara karşı önemli bir avantajıdır. Eksiklikleri negatif kenar içeren graph’larda kullanılamaması ve diğerlerine kıyasla daha yavaş olmasıdır. Çünkü bu algoritmada bir kere ziyaret edilen bir düğüm başka düğümler ziyaret edildikten sonra bir daha ziyaret edilemez. Bu yaklaşım ancak her kenarın pozitif değerli olduğu varsayımı kabul edilirse her zaman doğru sonucu verecektir.

Negatif değerli graph’larda bir kaynaktan diğer her düğüme olan en kısa yolları bulmak için Bellmann Ford Algoritması, bütün farklı kaynak-hedef kombinasyonları arasındaki en kısa yolları bulmak için Floyd-Warshall Algoritması kullanılır.

**A\* Algoritması**

A\* Algoritması, robotlarda kullanılan en kısa yol algoritmalarından bir tanesidir. ”Best First Search” algoritmasının hızını, Dijkstra en kısa yol algoritmasının kesinliğini sentezleyen bir algoritmadır. Çalışma prensibi olarak Dijkstra’ya çok benzerdir ancak aralarındaki fark, A\* algoritmasında kullanılan hedefe yönelmeyi kolaylaştıran “optimal cost to go” değeridir. Bu değer, algoritmanın 2 farklı yol ile karşılaştığında hedeflediği düğüme olan yolu görece daha kısa olarak algılamasını sağlar. Algoritmanın örnek bir graph için başlangıcı ve uygulamasında kullanılan “pseudo code” aşağıdaki gibidir.





Ancak karşılaştırmalara “optimal cost to go” değerini katmak, bir yandan da algoritmanın yanılabilirliğini arttırır. Kısacası en kesin sonuç için Dijikstra’yı, olması muhtemel verimli çözümleri bulmak için A\* kullanmak akıllıca olacaktır.

**Yol Planlama Algoritmaları**

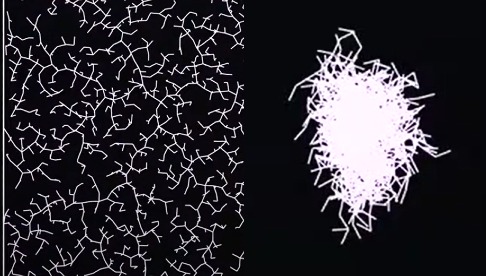
**C space, C obstacles, C free kavramları**

C space kavramı robotun bulunduğu uzayda bulunan diğer her şeyin bulunabileceği tüm noktaları içeren kümedir. C free, C space içinde robotun zarar görmeden hareket edebileceği noktaları ifade eden kümedir. C obstacle ise robotun “collusion” yani çarpışma yapmasına sebep olacak alanlardır. Aralarındaki ilişki: C space = C free + C obstacle küme toplamı şeklindedir.

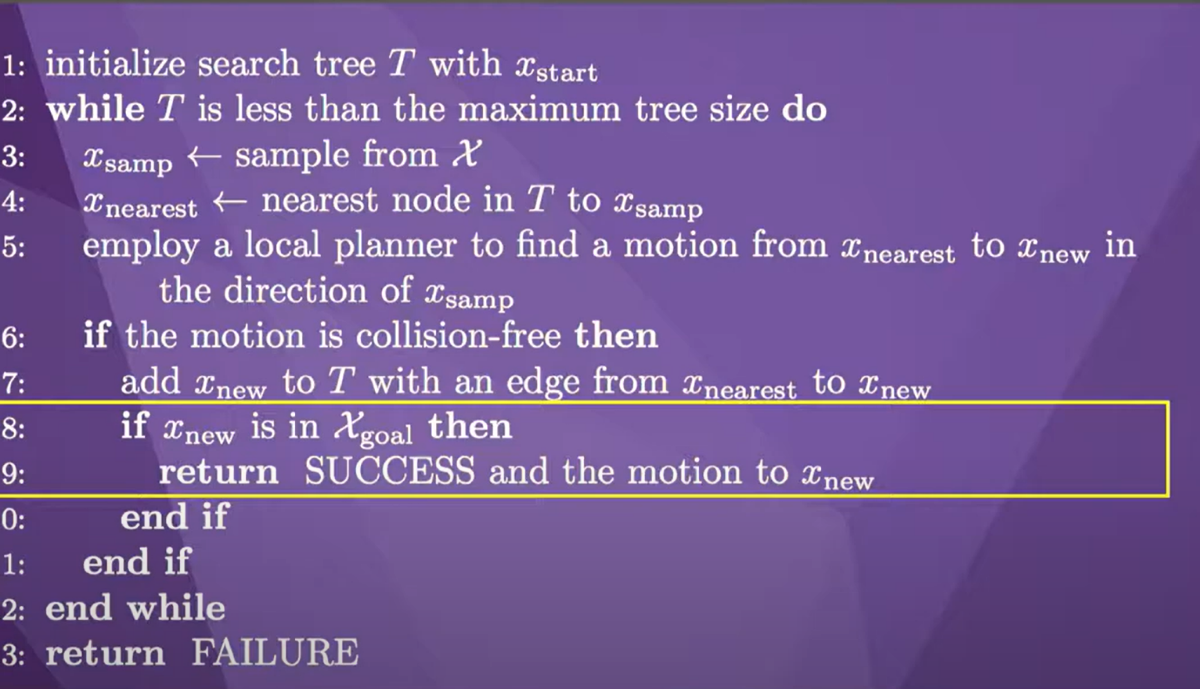
**RRT ve RRT\* (Rapidly Exploring Randomized Trees) Algoritmaları**

RRT Algoritması, belirtilen kaynaktan belirtilen hedef alanında bir düğüm oluşturulana kadar veya oluşturulan düğüm sayısı belirli bir sayıya ulaşana kadar C free içerisinde rastgele bir düğüm oluşturan ve oluşturduğu düğümü ona en yakın düğüme, tercih edilen “local\_path” algoritmasını kullanarak bağlayan bir yol bulma algoritmasıdır.

RRT Algoritması robotlar için bir taslaktır, pek çok “local\_path planner” ile çalıştırılabilir. RRT\* Algoritması, RRT Algoritmasını çalışma zamanı içerisinde yeniden yapılandırarak daha verimli hale getirmek için oluşturulmuştur. RRT metodu tıpkı bir cıvık mantar gibi saha üzerinde yayılmayı sağlar. Bu imkanı, “random walk” yani rastgele gezinti algoritması ile elde edemeyiz. Aşağıda Random Walk ve RRT algoritmaları karşılaştırılmıştır.



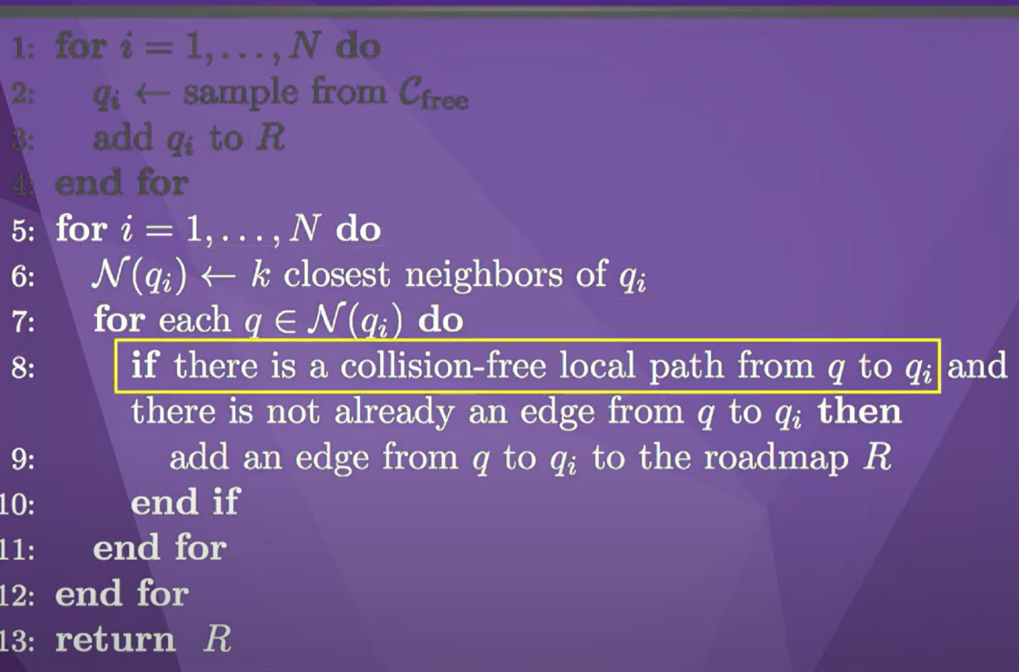
Algoritmanın akışında kullanılan “pseudo code” aşağıda verilmiştir.



**PRM Yol Planlama Algoritması**

“Probabilistic Road Map” olasılığa dayalı yol planlama algoritmasıdır. C space’te bulunan C free kümesi içerisinden belirtilen sayıda rastgele noktalar tanımlar. Ardından her noktayı belirtilen sayıda en yakın noktaya bağlayarak bir yol planlama ağacı oluşturur.

PRM yol planlama algoritması, C free üzerinden rastgele N tane nokta seçer, bu noktalarda düğümler oluşturur, oluşturulan her düğüm için o düğüme en yakın olan K tane düğümü o düğüm ile bağlar. Bu sayede robotun yol seçme –“execution of the plan”- işlemini yapabileceği, hedefe ulaşma olasılığı bulunan bir düğümler ağı oluşturulmuş olur. Algoritmanın doğru çıkma olasılığı, C free den seçilen düğüm sayısı arttıkça %100’e yaklaşır. Ancak hiçbir zaman doğru sonuç vereceği kesin değildir. PRM bize daha az node kullanarak amacımıza uygun bir yol haritası oluşturma imkanı sağlar, çok boyutlu C space problemlerini grid metotlarını kullanmadan çözümlememize yardımcı olur. PRM yol planlama algoritması için “pseudo code” aşağıda verilmiştir.

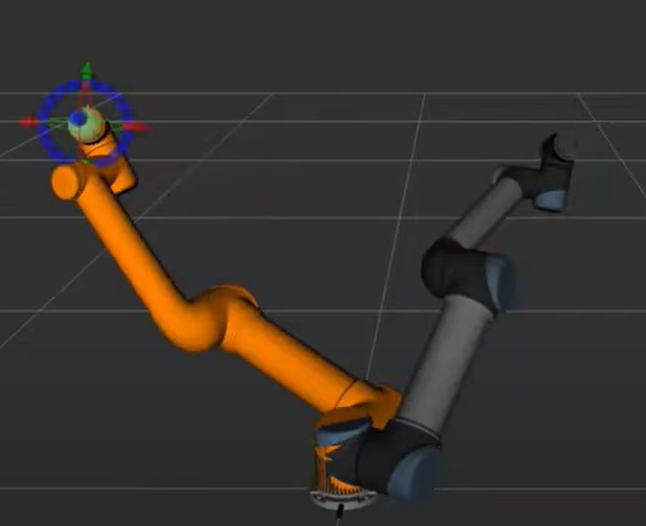


**Yol Seçme Algoritmalarının Optimizasyonu İçin Genetik Seçme Algoritması**

Araştırmam ile dolaylı olarak bağlı olmasına rağmen arama sürecimde ilgimi çeken bu konuya da kısa bir parantez açmak istiyorum.

Genetik seçme algoritması, ünlü Evrim Teorisyeni Charles Darwin’in kuzeni İngiliz Biyolog Francis Galton tarafından ortaya atılan “öjeni” kavramına dayanır. Öjeni, Yunanca’da “iyi doğmuş” anlamına gelmektedir. Öjeni kavramı, “İnsanoğlunun nesiller boyu gelişimini sürekli olarak sürdürmesi için en iyi özelliklere sahip olan insan kümesi dışındaki diğer insanların yapay olarak elenmesi, böylece nesiller devam ettikçe insan ırkının iyileşmesi” prensibine dayanır. Bu düşüncenin matematiksel olarak doğruluk payı olsa da her insanın yaşama hakkı olduğu düşünülürse bu fikrin etik dışı olduğu, bu sebeple insanlarda uygulanamayacağı anlaşılabilir. Ancak iyinin doğal olarak seçilmesine dayanan bu yöntem robot algoritmaları için gayettabii kullanılabilir.

Örneğin bir robotun elinin istenen konuma ulaşmak için kullandığı yol seçme algoritmasını ele alalım. Örnekte kullanacağımız robotun görseli aşağıda verilmiştir.



Gördüğümüz gibi robotumuzun 3’ü açısal olmak üzere 6 tane hareket boyutu var. bu boyutları (x1,y1,x2,y2,x3,y3) şeklinde bir şemada tanımlayalım ve bu şemaları “gen” olarak isimlendirelim. Yukarıdaki fotoğrafta turuncu ile gösterilen hede yola ulaşmak için PRM algoritmasını kullanarak rastgele genler oluşturalım. Her genin 6 parametresini de göz önüne alarak hedefe yakınlığını puanlayan bir fitness(f) fonksiyonu tanımlayalım. f fonksiyonunun çıktısına göre puanı yüksek olan genleri seçen bir de g fonksiyonumuz olsun. Artık genlerimizi oluşturduğumuza göre şimdi onları çaprazlayarak en iyi gen kombinasyonlarını elde etmeye çalışalım. Rastgele N tane gen oluşturalım ve bunların arasından iyilerini seçen h(x) = g o f(x) fonksiyonunu kullanalım. h fonksiyonunun görüntüleri arasında “crossover”, “mutasyon”, “çaprazlama” işlemlerini uygulayarak yeni gen kombinasyonları elde edelim. Elde ettiğimiz sonuçları belirli bir fitness alt değeri sağlanana kadar sürdürelim. Böylece, “öjeni” prensibine dayanan seçimimiz sonucunda robot yol planımızı optimize etmiş olduk.

Algoritmanın akışı kısaca anlatıldığı gibidir. Biraz daha detaylı incelemek için kaynakçaya göz atabilirsiniz.

**Sürü Zekası Çeşitlerinin Analizi ve Arama Kurtarma Çalışmaları İçin Kullanılması**

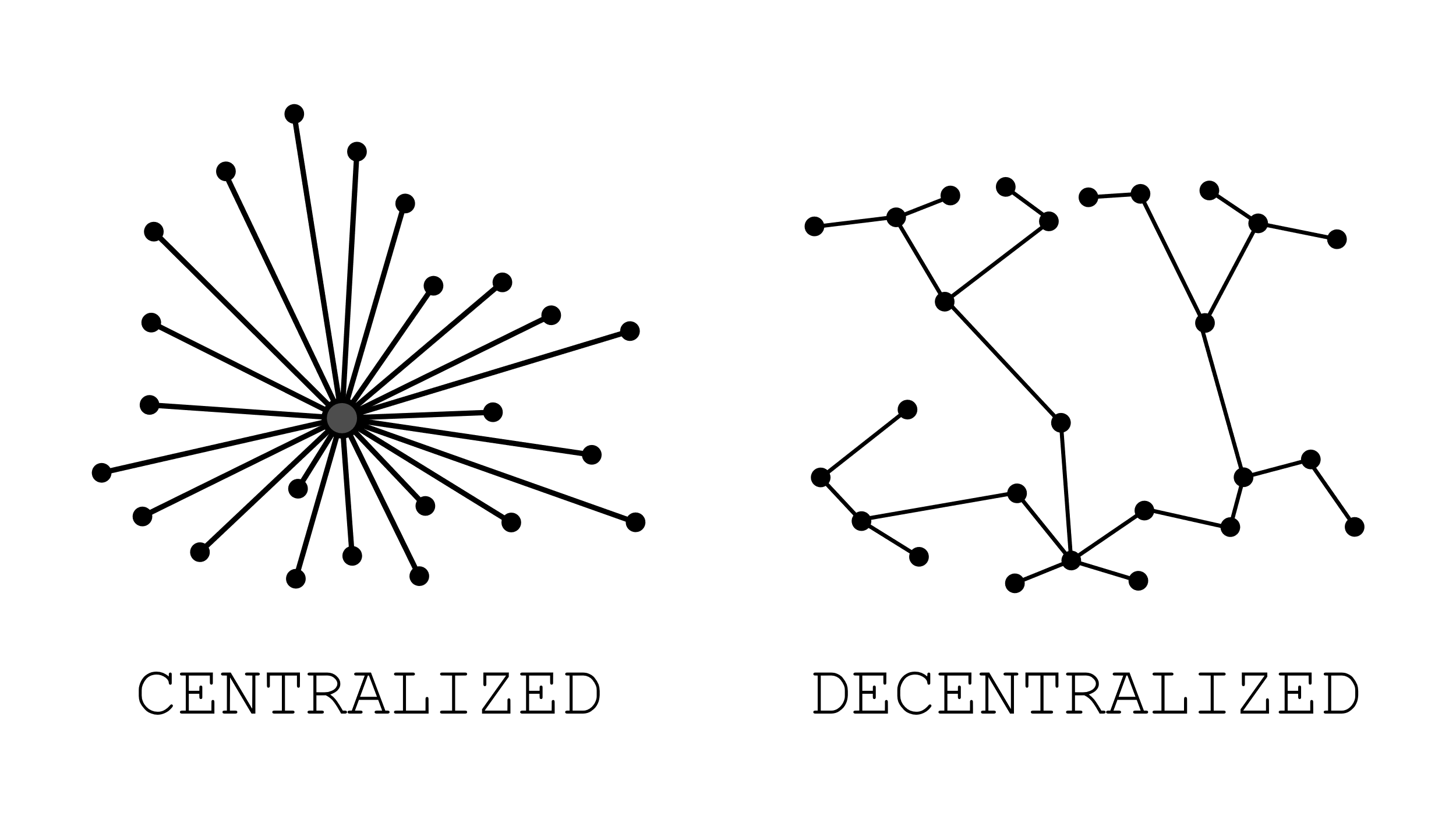
**Sürü Zekası ve Çeşitleri**

**Doğrudan İletişim**: Doğrudan İletişim, iki robotun birbirleri arasında aktarıcı başka bir kanal olmaksızın IR, özel frekanslardaki sesler gibi yöntemler kullanarak kurdukları iletişime denir.

**Dolaylı İletişim**: İki robotun birbirleri ile bir aktarıcı vasıtasıyla kurdukları iletişimdir. Burada aktarıcı bir robot olmak zorunda değildir, aktarıcının fonksiyonu bir robotun yaptığından diğer robotun kendisinin ne yapmasın gerektiğini çıkarsamasını sağlamaktır.

Örneğin, bir sürü zekası ile çalışan robotlarda bir robotun ağır yük taşıması için aldığı pozisyon, diğer robotun ona katılmak için haber almasını sağladığından 2 robot arasındaki aktarıcıdır.

İnceleyeceğimiz 2 ana sürü tipi aşağıdaki şemada kabaca gösterilmiştir.



**Sürü Zekası (Dağınık Sürü)**

Sürü zekası, kendi başına zayıf olan robot birimlerinin belirli bir iletişim ağı vasıtasıyla haberleşerek oluşturdukları kümilatif ortak zekadır. Bu zeka tipinde bir birim olarak robotların sahip olmakla yükümlü olduğu özellikler: Kendi sensör girdilerini almak, bu girdileri bir başka robota aktarabilmek ve alınan ortak karara uygun hareket edebilmektir.

Basit yapısı olan birçok robot kullanmak, tek bir güçlü robot kullanmaya kıyasla oldukça avantajlıdır: Üretimleri kolaydır ve seri üretilmeye müsaittirler, esnektirler: grubun bir üyesinin zarar görmesi diğer üyelerin çalışmasını etkilemez -oysa bu, tek başına çalışan kompleks bir robotta tüm sistemin çökmesine sebep olabilir-, Tek başına çalışan bir robota göre çok daha yüksek bir arama kapasitesine sahiptirler, bundan dolayı uzay araştırmalarında sıklıkla kullanılırlar.

Ancak sürü zekasının elbette görmezden gelinemeyecek dezavantajları da vardır: Sürü zekası ile çalışan robotlar, robotlardan aldığı verilere dayanarak son kararı verecek merkezi bir robot olmadığından kolaylıkla kandırılabilirler, bu robotlarda kullanılan mesaj ağı kandırılmanın mümkün olduğu durumlarda bir merkez tarafından denetlenmelidir. Ayrıca bu robotlarda görev dağılımı yapmak her robot aynı statüye sahip olduğu ve onlara bu görevleri atayacak özel bir birim olmadığı için zor bir hale gelir.

Sürü zekası, günlük hayatta pek çok alanda karşımıza çıkmaktadır: Fabrikalarda ürünlerin robotlar aracılığıyla taşınması, astronomide yeni alanların keşfedilmesi, insan vücudunda kanser hücrelerinin tespiti ve imhası, madenlerde cevher bulma keşifleri, afet sonrası arama kurtarma gibi pek çok alanda kullanılmaktadır.

**Merkezi Sürü**

Merkezi sürü robot birimlerinin kararlarının belirli bir merkez tarafından yönlendirildiği robot sürü sistemidir. Tüm robotlar; hareket etmek için verilecek komutları, hareketi etkileyecek bileşenleri ilettikleri bir sistemden alırlar. Biyolojik bir benzetme yapacak olursak: robotlar birer uzvumuz, onları yöneten merkez ise beynimizdir. *Bu sisteme benzeyen bir başka sistem Birçok alt merkezin iletişimine dayanan hiyerarşik sürü sistemidir.*

Merkezi sürüler, robotların harfiyen uygulamaları gereken komutlar aldıkları sistemlerde kullanışlıdır. Alınan veriler bir merkez tarafından değerlendirilerek uygulandığı için avantajlıdırlar. Ancak verilen komutlar sadece bir merkeze bağlı olduğundan sistemdeki veri akışı, dağınık sürünün aksine bir birime bağlıdır. Bu birimin imha edilmesi durumunda tüm sistem çökmeye müsaittir.

**Hibrit Sürüler**

Hibrit sürüler, az önce bahsettiğimiz 2 sürü tipinin özelliklerini sentezleyen melez bir sürü tipidir. Bu tip sürülerde, robotlar kendi içlerinde gruplara ayrılırlar ve her grup kendi içinde özerktir, bir grup bir mesaj algıladığında bu mesaja o grubun üyeleri cevap verir, diğer üyeler için bu mesajlar geçersizdir. Bu sistemi ünlü Komutan Napoleon Bonapart’ın Bölük sistemine (Corp System) benzetebiliriz.



**Afet Sonrası Arama Kurtarma ve Bu Süreçte Sürüler Arası İletişim**

Öncelikle afet dediğimizde aklımıza sadece deprem gelmemeli, bir olayın afet olarak değerlendirilmesi için insanlarda can ve mal kaybına yol açması yeterlidir. Örneğin maden çökmeleri, uçak kazaları da afete birer örnektir.

Afet sonrası arama kurtarmanın önemini hepimiz biliyoruz. Peki, arama kurtarmada insanlar yerine robotları kullanmanın avantajları nelerdir? Öncelikle robotlar bir ortamda verimli çalışabilmek için pil, şarj gibi kolay sağlanabilir şeylere ihtiyaç duyarlarken insanlar; aydınlık, solunabilir hava içeren alanlara ihtiyaç duyarlar. Ayrıca robotların çalışma güçleri insanlara göre çok yüksektir: hiç 2000 kiloluk enkazı sırtlayan bir insan gördünüz mü? Son olarak robotlar arası iletişim insanlar arası iletişimden çok daha hızlıdır.

Afet sonrası kurtarmada en önemli bileşen, yardımların en hızlı şekilde yapılmasıdır. Bunun için de robot sistemleri kendi içlerinde ve kendi aralarında en iyi şekilde iletişim kurmalıdırlar. Raporumu bu ilişkiyi örnekleyen bir model sunarak bitireceğim.

**Afet Sonrası Arama Kurtarma İçin Örnek Sistem**

Sistemimiz; su altı, kara ve hava olmak üzere 3 farklı insansız araç tipinden oluşuyor. Amacımız deprem olan ve depremin tsunamiyi tetiklediği bir afet bölgesinde arama kurtarma işlemini en hızlı şekilde yapmak.

**İnsansız Kara Araçları**: Bu araçlar; tsunaminin karaya zarar vermesini önlemek ve karada ilk yardıma, yiyecek ve suya ihtiyaç duyan insanlara malzeme sağlamak için kullanılacak hiyerarşik bir sürü olacak.

**İnsansız Su Altı Araçları**: Bu araçlar, depremlerin tespitinde önemli yer tutan su altı sensörlerinden bilgi alıp bu bilgileri oy çokluğuna dayanarak toplu olarak 4 magnitüd ve üzeri deprem oluyor ya da olmuyor olarak yorumlayıp merkezi iletic birime iletecek.

**İnsansız Hava Araçları**: Bu Araçlar, Enkaz ya da su altında kalmış insanları algılamak için IR ya da kalp atışı denetleme sensörleri kullanacak, keşif için özel bir algoritma kullanacak ve yaptığı keşifleri doğruluk yüzdesine göre önceleyerek sisteme bildirecek dağınık bir sürü olacak.

**Merkezi İletici Birim:**

Aldığı verilere bağlı olarak gereken emirleri diğer robotlara aktaracak olan otonom bir sistem olacak.

**Arama Kurtarma Ekibi (İnsanlar):** Robotların yardımı ile arama kurtarma işlemini gerçekleştirecek olan insanlardan oluşan grup.

**Sistemin Akışı ve Çalışma Şekli:**

**Sarsıntı algılandığında:**

1. Su altı sensörleri sarsıntıyı algılar ve kendi içlerinde oylama yaparak deprem durumunu merkezi ileticiye bildirir.
2. Merkezi iletici, insansız hava araçlarına havalanarak arama yapmaları için emir verir.

Not: Bu arama için tüm hava araçlarının aktif olması, dağınık sürü mantığına göre çalıştıkları için gerekmez.

1. İnsansız kara araçları okyanusun 5 sn lik videosunu çeker ve “makine öğrenmesi” algoritmalarına dayanarak tsunami algılandığı durumda şehir bariyerlerinin yükseltilmesi için uygulamaya geçer.
2. İnsansız hava araçları, alanı inceler ve afetzede tespit ettikleri alanları merkezi ileticiye bildirirler ve bu alanda spiral çizerek keşiflerine devam ederler. Araştırmalar sonucu ulaşılan veriler, bu yöntemin afetzedelere hızlıca ulaşmak için en verimli yol duğunu göstermektedir(bkz. 7. kaynak)
3. Merkezi iletici, insansız kara araçlarına yiyecek, ilk yardım malzemesi ve suyu gerekli alana götürmeleri için bu alanların bilgilerini verir.
4. Bilgiyi alan insansız kara araçları, enkazı kaldırmak ve gereken diğer ihtiyaçları asağlamak için arama kurtarma ekibi ile ilgili alanlara giderler.
5. İnsansız kara araçları yerleri bildirilen afetzedeleri, enkazdan kaldırmak için öğrendikleri “makine öğrenmesi” algoritmalarını kullanılırlar.
6. Enkazın dengeli bir şekilde tutulması ile enkazın altından çıkarılan depremzedelere arama kurtarma ekibi ilk yardım yapar ve onları psikolojik olarak sakinleştirir.
7. 4 ve 8. Adım arasındaki döngü belirlenen afetzedelerden yaşadığı algılananların sayısı 0’a inene kadar eder.

İnsansız kara, hava, su altı araçları ve insanlar arasındaki koordinasyonun afet sonrası arama kurtarmada etkin olarak kullanılmasını amaçlayan bu hayali sistem ile raporumu sonlandırıyorum.

**Kaynakça**

Birinci Kaynak: EinarUeland. (n.d.). GitHub - EinarUeland/Astar-Algorithm: A\* (Astar / A Star) search algorithm. Easy to use. GitHub. https://github.com/EinarUeland/Astar-Algorithm

İkinci Kaynak: Mburst. (n.d.). GitHub - mburst/dijkstras-algorithm: Implementations of Dijkstra’s shortest path algorithm in different languages. GitHub. https://github.com/mburst/dijkstras-algorithm

Üçüncü Kaynak: KaleabTessera. (n.d.). GitHub - KaleabTessera/PRM-Path-Planning: Implementation of Probabilistic Roadmap Path Planning Algorithm. GitHub. https://github.com/KaleabTessera/PRM-Path-Planning

Dördüncü Kaynak: Ahnafraihan. (n.d.). GitHub - ahnafraihan/BestFirstSearch: C++ implementation of the Best-First Search algorithm to solve the 0-1 Knapsack Problem. GitHub. https://github.com/ahnafraihan/BestFirstSearch

Beşinci Kaynak: Vvrs. (n.d.). GitHub - vvrs/RRT\_Algorithms: Implementation of Bi-directional RRT\*. GitHub. <https://github.com/vvrs/RRT_Algorithms>

Altıncı Kaynak: Adams AL, Schmidt TA, Newgard CD, Federiuk CS, Christie M, Scorvo S, DeFreest M (2007) Search is a time-critical event: when search and rescue missions may become futile. Wilderness and Environmental Medicine 18(2):95–101

Yedinci Kaynak: Arnold, R., Yamaguchi, H., & Tanaka, T. (2018). Search and rescue with autonomous flying robots through behavior-based cooperative intelligence. Journal of International Humanitarian Action, 3(1). https://doi.org/10.1186/s41018-018-0045-4s