

**İSTANBUL OKAN ÜNİVERSİTESİ**

**MESLEK YÜKSEKOKULU**

**BİLGİSAYAR TEKNOLOJİLERİ BÖLÜMÜ**

**YAPAY ZEKÂ**

**MİNİMUM YOLUN BULUNMASI**

**PROJE RAPORU**

**BÖLÜM**

**BiLGİSAYAR PROGRAMCILIĞI**

**NUMARA**

**18MY03015**

**PROJEYİ HAZIRLAYAN ADI-SOYADI**

**EMİRHAN KARAPINAR**

**DANIŞMAN/ÖĞRETMEN**

**NİLGÜN İNCEREİS**

**İÇİNDEKİLER**

Minimum Yolun Bulunması-------------------------------------------------------------------3

Giriş----------------------------------------------------------------------------------------------------------3-4

Genel Bilgiler-----------------------------------------------------------------------------------------------4

Problem Formülasyonu Ve Karmaşıklık-------------------------------------------------------------4

Örnekleme-------------------------------------------------------------------------------------------------5

Sat Modeli ------------------------------------------------------------------------------------------------5

Akış Modeli -----------------------------------------------------------------------------------------------5

Ulaşabilirlik Modeli -------------------------------------------------------------------------------------6

ASP Modeli ---------------------------------------------------------------------------------------------6-7

MiniZinc Modeli -----------------------------------------------------------------------------------------7

Deneysel Değerlendirme ------------------------------------------------------------------------------8

Eğrilik Bilgilerini Yakalama --------------------------------------------------------------------------8-9

Birden Fazla Hücre İle Çalışma -----------------------------------------------------------------------9

Algoritma --------------------------------------------------------------------------------------------------9

Sezgisel Dinamik Programlama ---------------------------------------------------------------------10

Sonuçlar ----------------------------------------------------------------------------------------------10-11

Proje Bilgileri -----------------------------------------------------------------------------------12

Proje Konusu---------------------------------------------------------------------------------------------12

Algoritma Yapısı-----------------------------------------------------------------------------------------12

Graf Yapısı-------------------------------------------------------------------------------------------------13

Akış Şeması-----------------------------------------------------------------------------------------------13

Öz Geçmiş------------------------------------------------------------------------------------------------14

Kaynakça-------------------------------------------------------------------------------------------------14

Ek-------------------------------------------------------------------------------------------------15

Projemi Nasıl Yaptım-------------------------------------------------------------------------16

**MİNMUM YOLUN BULUNMASI**

**GİRİŞ**

Elektrikli araçlara (EV'ler) ve Plug-in Hybrid Elektrikli Araçlar (PHEV) gibi varyantlarına olan ilgi, ekonomik, çevresel ve benzinle ilgili güvenlik endişeleri. Bir PHEV,karşılaştırılabilir bir konvansiyonel araç (CV) ile ilgili olarak hem taşıma maliyetleri hem de sera gazı emisyonları (Windecker ve Ruder 2013). Bir elektrik var motor ve güç kaynakları olarak bir içten yanmalı motor (ICE). Vardır bir EV'nin normal bir elektrik prizinden şarj etme ve Uzun menzilli yolculuklar gibi benzinle çalışan bir CV'nin rahatlığı. Şarj sürdürme (CS) modunda, tek enerji kaynağı olarak benzin kullanarak seyahat eder. Şarjda(CD) modunu tüketen PHEV'ler yalnızca elektrikle seyahat edebilir veya hem elektrik hem de benzin (Pistoia 2010, Axsen ve Kurani 2010, Axsen ve ark.2008, Markel ve Wipke 2001). Karışık bir tarzda, PHEV öncelikle elektrik motoru kullanarak, ICE tarafından desteklenen benzinli işlemler için ekstra güç gerektirir. Son araştırmalarda tümüyle elektrikli CD modu sürücüsü varsayılmıştır Traut ve ark. (2011) ve He ve ark. (2013). Benzer şekilde, bu makalede,CD modunda yalnızca elektrik kullanarak çalışan PHEV'lere odaklanın. Ancak,önerilen metodoloji ayrıca, bunlar için yakın bir yaklaşım olarak görülebilir.Birincil enerji kaynağı tekrar olduğu için harmanlanmış modda çalışan PHEV'ler elektrik ve ICE sadece ek olarak kullanılır.

Özerk bir araç için bir yol planlama sorunu belirli bir ortamda, engellerden kaçınırken,birkaç yıl çalışmış .Ayrıntılı anketler yol planlama ve hareket planlama algoritmaları sağlanır,örneğin, Latombe, Hwang ve Ahuja ve daha yakın zamanda, LaValle. En genel olarak,yol planlama sorunu verilen bir başlangıçtan bir yol bulmaktır.Ortamdaki belirli bir hedefi gösterecek şekilde herhangi bir engelle kesişmez ve böylece yolu araç takip edebilir. Zaman parametrelendirmesi yol boyunca izlenmesi gereken bir yörüngeye yol açar araç. Son adım sırasında, yörüngeye uymak gerekir ilgili araç dinamikleri tarafından getirilen kısıtlamalar,belirli bir maliyet işlevini de en aza indirirken, ör. Zamanı seyahat. Uygun bir yörüngenin hesaplanması hedefin başlangıç ​​noktasına hareket denir planlama sorunu.

Bariz iyimserlik eksikliğinin yanı sıra, yaklaşım geometrik ve dinamik kısımlarının ayrı ayrı ele alınması sorun ayrıca dinamik olarak mümkün olmayan yollara da yol açabilir;nedeni geometrik yol planlayıcının taşıtın dinamik sınırlamaları hakkında ön bilgi.Genel şemanın her zaman uygulanabilir yollar oluşturmak, arasındaki boşluğu kapatmak zorundayız geometrik ve dinamik katmanlar. Bu sadece elde edilebilir dinamik zarfı hakkında belirli bilgiler varsa araç yol planlayıcıya aktarılır.Bu yazıda, dahil edilecek yeni bir plan öneriyoruz dinamik olarak uygulanabilir yolların sınıfı hakkında bilgi erken (viz geometrik tabakada). Ayrıca,Bu, sayısal olarak verimli bir şekilde Dijkstra’nın algoritması için önemsiz bir değişiklik grafikler üzerinde en kısa yol problemlerinin çözümü. Bu anlamda, algoritmamız sadece araçtan daha genel ilgi görüyor yol / hareket planlaması. En kısa aramak için kullanılabilir düğüm geçiş maliyetleri ne zaman bağımlıysa, grafikteki yollar ziyaret edilen düğümlerin geçmiş tarihinde. Basit örneklerle, önerilen yaklaşımın başlangıçta bir yol planlamanın standart kısa görüşlü stratejisi dinamik zarfı dikkate almadan Araç.

Hücre ayrışmasına dayanan geometrik yol planlama yöntemleri engelsiz yapılandırma alanını dışbükey, üst üste binmeyen bölgeler, hücreler olarak adlandırılır ve ardından Dijkstra’nın algoritması gibi bitişik hücreler dizisi için bağlantı grafiği hedefe başlangıç ​​noktası. Biri en yaygın olarak kullanılan yaklaşık hücre ayrıştırma teknikleri dörtlü yöntemdir. Çoklu çözünürlük değişen lokal / global hücre ayrışımlarını kullanan şemalar ince / kaba çözünürlük. Diğer hiyerarşik yol planlama teknikleri, erken yinelemelerde yolların MIXED hücreleri arasında dolaşmasına izin ver ve daha sonra bu yinelemelerde bu yolları daraltın, böylece sadece ÜCRETSİZ veya TAM hücreleri içerir.

**GENEL BİLGİLER**

**Problem Formülasyonu ve Karmaşıklık**

Çok kaynaklı-çoklu-hedef (MOMD) yol bulma problemi aşağıdaki gibi formüle edilir. Yönlendirilmiş bir ark ağırlıklı grafik G = (V, E, w) varsayalım, burada V bir köşe noktası kümesidir, E bir dizi yönlendirilmiş yay kümesidir ve w: E → N bir yayların negatif olmayan ağırlıklara eşlenmesi. P bir dizi olsun paketler, böylece her p ∈ P paketi bir çift olarak tanımlanır (orig (p), dest (p)), burada orig (p) ∈ V p'nin orijinal konumudur ve dest (p) ∈ V p'nin hedef konumudur. Görev her bir paket için ark alt altlığı A-E seçmek için p, G0 = (V, A) grafiğinde orig (p) 'den dest (p)' ye ve A (ki) 'deki yayların ağırlıklarının toplamı P a wA w (a)) minimaldir.Giriş bölümünde belirtildiği gibi, MOMD problemini optimal olmayan bir şekilde çözmek için basit bir yöntem vardır.İlk olarak, her paket için en kısa yolu bulun, örneğin FloydWarshall algoritması gibi bir çift-en kısa yol algoritması kullanan [2]. Sp (p) tarafından kullanılan yay seti olsun paket için en kısa yol s. Ardından A'yı ∪p∈P sp (p) olarak tanımlayın.Açıkçası, bu yöntemin polinom zaman karmaşıklığı vardır (Floyd-Warshall algoritması zaman karmaşıklığına sahiptir O (| V |3) ve daha hızlı yöntemler vardır, örneğin,grafik seyrek veya paket sayısı az Dijkstra’nın algoritması daha iyi bir seçenektir). Ancak, bu yöntem bölümünde göstereceğimiz gibi hiçbir garanti optimallik yok deney sonuçları ile. Bu yazıda çözüme odaklanıyoruz MOMD problemi en uygun şekilde; hiçbirinin farkında değiliz sorun için bilinen yöntem.Önce MOMD sorununun NP zor olduğunu gösterelim sorun. Sorunun karar varyantını varsayalım, görevin verilen herhangi bir k sayısının bir dizi ark P E vardır, öyle ki P a∈A w (a) ≤ k ve her paket p için grafikte yönlendirilmiş bir yol vardır G0 = orij (p) 'den dest (p)' ye kadar (V, A). Açıktır ki MOMD sorununun karar varyantı NP'ye aittir karmaşıklık sınıfı, çünkü A kümesi verildiğinde, doğrulanması kolaydır.Polinom zamanında, A gerçekten de MOMD için bir çözümdür azaltılmış her paket için bir yol bularak sorun grafik G0 = (V, A).MOMD sorunu iyi bilinen minimum değerlere benzer Karp’ın orijinal 21'inden biri olan Steiner ağacı sorunu NP-tam problemleri. Aşağıdaki genel minimum Steiner ağacı probleminin sürümü. Bize veriliyor kenar ağırlıklı bir grafik G = (V, E, w) ve bir alt küme S ⊆ V gerekli köşelerin. Steiner ağacı G'de genişleyen bir ağaçtır.S'nin tüm köşeleri Sorunun iki çeşidi vardır:optimizasyon problemi, görev minimum ağırlık bulmaktır.Steiner ağacı; karar probleminde bize bir k değeri verilir ve görev, toplam ağırlıktaki bir Steiner ağacının en fazla k var.

**ÖRNEKLEME**

Pratik olarak önemli ve teorik olarak zorlu bir sorun sunuyoruz: Yakıt ikmali ve akü anahtarlama istasyonları bulunan bir şebekedeki geçmeli hibrid elektrikli araçlar (PHEV'ler) için minimum maliyet yolunu bulmak, elektrik ve farklı maliyet yapıları ve sınırlamaları olan enerji kaynakları olarak benzin. Biz elektrikli araç ve konvansiyonel araç özel durumları polinom olarak çözülebilir olsa da bu sorunun NP-tamamlanmış olduğunu göstermektedir. Üç çözüm öneriyoruz teknikler: (1) araç amortismanı, akü degradasyonu ve durdurması gibi yakıt dışı maliyetleri içeren karışık tamsayı kuadratik olarak kısıtlanmış bir program,(2) dinamik programlama tabanlı buluşsal yöntem ve (3) en kısa yol buluşsal yöntemi. Biz hem gerçek dünya karayolu ağını kullanarak kapsamlı hesaplama deneyleri yapın veri ve çeşitli boyutlarda yapay olarak üretilen yol ağları ve sürücü tercihlerinin etkileri ve pilin kullanılabilirliği hakkında önemli bilgiler sağlar PHEV ekonomisi üzerindeki anahtarlama istasyonları. Özellikle, bulgularımız akü değiştirme istasyonlarının sayısının arttırılması sürücülerin menzil kaygısının üstesinden gelmek için yeterli olmayabilir.Anahtar Kelimeler: plug-in hibrid elektrikli araçlar, minimum maliyet yolu, araç güzergahı,enerji yönetimi, tamsayılı programlama, dinamik programlama.

**SAT Modelleri**

MOMD problemi için iki kısıtlama modeli tanımladık daha düşük maliyetle kısıtlanmış bir çözüm bulmak için kullanılabilir ve üst sınırlar. Bu modeller daha sonra bir dikotomik dal ve sınır algoritmasını kullanarak optimum çözüm. Boolean değişkenleri üzerindeki aritmetik kısıtlamaları kullanarak kısıtlama modelini tanımlayacağız ve bu model daha sonra bir SAT formülüne çevrildi.

**Akış Modeli**

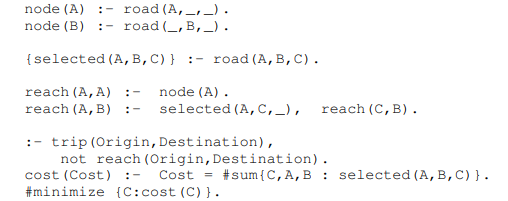
MOMD sorunu, her paket için paketin kaynağından paketin hedefine bir yol olmasını gerektirir. Flow modeli, her biri için ayrı bir yol tanımlar paket, paketler tarafından kullanılan tüm yayları biriktirir ve toplam maliyeti kısıtlar.G = (V, E, w) yönlendirilmiş bir ark ağırlıklı grafik olsun ve P paketler kümesi olmak. Her paket için p ∈ P ve her ark için a ∈ E bir Boole karar değişkeni sunuyoruz İkinci el [a, p] p paketini taşımak için ark a'nın kullanılıp kullanılmadığını gösterir p. Her paket için p ∈ P ve her köşe için x ∈ V bir Boole değişkeni Akış [x, p],p paketinin taşınması x köşesinden geçer.InArcs (x), x ve OutArcs (x), x'den giden yaylar kümesi olabilir. Resmi olarak InArcs (x) = {(y, x) | (y, x) ∈ E}, OutArcs (x) = {(x, y) | (x, y) ∈ E}Bir paketin taşıma yolunu modellemek için akışı belirleriz koruma kısıtlamaları. Bu kısıtlamalar,paket kökeni terk etmeli ve hedefine ulaşmalıdır ve paket bir tepe noktasından geçiyorsa tepe noktasına girin ve bırakın (her ikisi de tam olarak bir kez). Durumda menşei, paket sadece onu bırakır ve benzer şekilde,hedef, paket sadece girer.

**Ulaşılabilirlik Modeli**

Flow modeli her paket için açıkça bir yol bulur. Gibi bazı paketler yollarının bir bölümünü paylaşabilir,Köşeler arasında sadece bir kez bir yol bulmak yararlıdır her paket için bir yol olmasını sağlar. Bu fikir, aralarında bir yol varsa (seçilen yayları kullanarak) herhangi bir köşe noktasına karar veren Okunabilirlik modelimizin arkasındadır.bir tek). Açıklandığı gibi aynı G ve P girişini varsayacağızYukarıda.Her bir köşe çifti için model, bunlar arasında bir yol olup olmadığını açıklar köşe noktası. Model temelde Floyd-Warshall'ı taklit ediyor algoritma . Yine Boole karar değişkenlerini kullanıyoruz Çözeltide belirli bir ark a ∈ E'nin seçilip seçilmediğini göstermek için [a] kullanılır. Köşelerin dizine eklendiğini varsayın (tamamen sıralanmıştır) {1, 2,. . . , | V |} (“x düğümü” kullanıldığında indeksi anlamına gelecektir).Boole karar değişkenleri Erişim [x, y, z] veya yalnızca i köşelerini kullanan x'ten y'ye bir yol var mı öyle ki i ≤ z. Özellikle Reach [x, y, 0] bir yay olduğunu söylüyor x'den y'ye var ve çözeltide veya x = y'de kullanılır.Değişkenler aşağıdaki kısıtlamalar kullanılarak bağlanır:∀x, y ∈ V: Erişim [x, y, 0] =1 x = y ise,0 (x, y) ∈ / E ise,A = (x, y) ∈ E ise [a] kullanılır.∀x, y, z ∈ V:[X, y, z] = maks. Ulaş ([x, y, z - 1],Erişim [x, z, z - 1] × Erişim [z, y, z - 1])Şimdi, her p paketi için,dest (p) 'ye orig (p) vardır:∀p ∈ P: [orig (p), dest (p), | V |] = 1 değerine ulaşın Amaç işlevi kısıt (1) kullanılarak ifade edilir.Flow modelinde olduğu gibi. Şimdi, Boole kararının sayısı değişkenler (n + 1) n 2 + e. Numaraya dikkat edin paket sayısına bağlıdır.

**ASP Modeli**

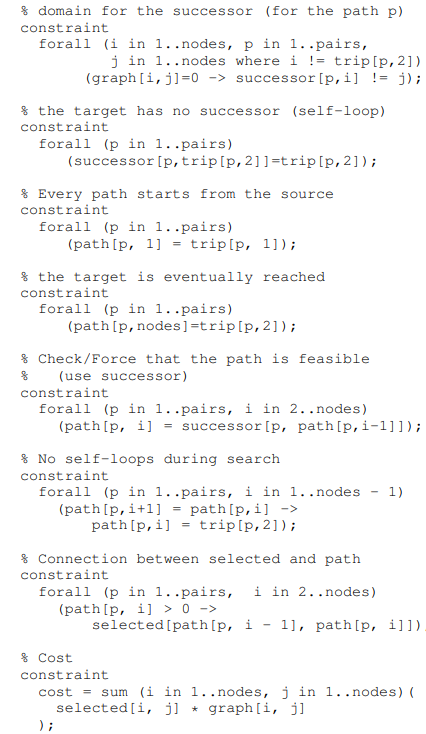
Yanıt Seti Programlama (ASP) içindeki MOMD sorununu kodladık. ASP'de her zamanki gibi kodlama çok özlüdür ve temel olarak bir oluşturma ve test programlamasına dayanır düzeni. Girdi grafiğinin bir yüklemle kodlandığını varsayın yol / 3 burada yol (a, b, w) yayın (a, b) maliyet w. Her başlangıç ​​/ varış noktası (o / d) çifti, ikili bir yüklem tarafından uygulanır. Yüklem düğümü şu şekilde tanımlayabiliriz: projeksiyon. Ardından, bir seçim kuralı kullanarak bir kenar veya değil. Okunabilirlik,seçilen düğümler ve tüm o / d çiftleri arasındaki bağlantıyı zorlamak için bir kısıtlama eklenir. Son olarak, maliyet optimizasyonu dayatılan.



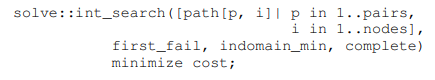
Tüm grafikteki her bir o / d çifti için minimum yol olabilir sınırları hesaplamak için birkaç kod satırı ile modellenmiştir birbirini izleyen aramada kullanılabilir. Elde edilen kod ancak büyük topraklama sorunu yaşar ve çalışma süresi (biraz) yukarıdaki koddakinden daha az örnekleri.

**MiniZinc Modeli**

MiniZinc'te de benzer, son derece deterministik olmayan bir kodlama test ettik [6]. Temel olarak, her bir o / d çifti için (p vardır) biz bir dizi yolu [i] tanıtmak nerede i = 1,. . . , p seçilen kenarları kullanarak yolu saklamaya yöneliktir.Bu dizilerin uzunluğu n'dir (düğüm sayısı). [i, j] yolu i-th başlangıcını yönlendiren yolda bulunan j-düğümü i-th hedefi. Hedef d'ye ulaşıldığında vektörün tüm ardışık değerleri d'dir. Başka hiçbir “döngü” mümkün değildir. Ayrıca yardımcı bir halef kullanmak için uygun bulduk yolun sonraki öğesinin etki alanını kısıtlamaya izin veren yüklem.



Çalışma süreleri çok tatmin edici değildir. Bu program tasarlanmış arama ile deneysel olarak en iyi sonuçları bulduk Sezgisel sonuçlar.



**Deneysel Değerlendirme**

Picat'ta SAT modellerini uyguladık.Lingeling'i SAT çözücü olarak kullanıyor. Picat'ta aynı modeller için CP ve MIP çözücüleri kullanmak da mümkündür, ancak SAT çözücü, CP ve MIP'den çok daha iyi performans gösteriyor.Bu modeller için çözücüler. Picat 1.9b1 ile birlikte kullandık 8 GB 1600 ile 1.7 GHz Intel Core I7 üzerinde MacOS X 10.11.4 MHz DDR3 RAM. Bu karşılaştırma için örnekleri kullandık Uluslararası Planlama Yarışması 2014'ün optimal parkurundan Ulaşım alanının adı . Tablo 1, alt sınır, üst sınır ve her bir örnek için en uygun çözüm.Tablo 1 ayrıca her iki SAT için çalışma zamanlarını (saniye cinsinden) gösterir en uygun çözümleri bulmak ve kanıtlamak için modeller. Akış modeli tüm test örneklerinde açıkça daha hızlıdır. Bu muhtemelen Flow modelinin daha az karar kullanmasından kaynaklanmaktadır.Örnekleri için Okunabilirlik modelinden farklı değişkenler.Ayrıca ASP ve MiniZinc'te sorun, her ikisi de erişilebilirlik kısıtlamaları kullanır.Varsayılan ayarlı clingo çözücü [7] 20 vakanın 19'u için 12'si bulunan en uygun çözümleri buldu her biri 1 dakika içinde, ancak daha fazla zaman aldı modellerimiz çözüldü ve p17 örneğini 24 saat içinde çözemedi. ASP modeli ve SAT modellerimiz SAT kullanıyor, modellerimiz önemli ölçüde daha fazla kısıtlamalar için kullanılan kompakt ve verimli kodlamalar nedeniyle ASP'den daha verimlidir.Ayrıca modellerimizi uygulanan bir modelle karşılaştırdık döngüleri önlemek için kısıtlamalar kullanan MiniZinc ‘da.Gecode tarafından çalıştırıldığında bu model, 7'si her biri 1 dakika içinde çözülen 20 vakanın 10'unu çözdü, ancak 24 zaman sınırı altındaki örneklerin 10'unu çözemedi örnek başına saat. Bu karşılaştırma bir kez daha SAT modellerinin problem için etkinliği.

**A- Eğrilik Bilgilerini Yakalama**

Engelsiz alanın hücre ayrışmasına dayanan mevcut yol planlama algoritmaları,kinematik veya dinamik kısıtlama olmadığında yollar oluşturma mevcut. Ancak, gerçek aracın hareketi bu tür kısıtlamalara uymak zorundadır. Ek varsayımlar olmadan,uygulanabilir bir yörüngenin tatmin edici olduğuna dair bir garanti yoktur.bu kısıtlamalar hücre kanalı içinde bile var olacak yol planlama algoritması ile hesaplanır. İlk bakışta, bir bunun sadece uygunsuz bir eser olduğunu iddia edebilir ilgili grafikte kenar maliyeti işlevinin seçimi. Aşağıda bu argümana karşı bir örnek sunuyoruz.Şekil 1'de gösterilen yol planlama problemini düşünün,burada S başlangıç ​​pozisyonunu, G hedefi belirtir ve karanlık alanlar engeldir. İki araç A ve B'yi düşünün, minimum dönüş yarıçapları kinematik olarak kısıtlanmış.min ve rB min sırasıyla, öyle ki rA min ≤ / 2 ve rB dk>. Açıkçası, Şekil 1'deki kesikli yol araç için uygundur A, ancak B aracı için değil. B için bir yol planlama algoritması Şekil l'de gösterilen kalın yolla sonuçlanmalıdır.Şekil 2 (a) aynı sorunu tek tip bir hücre ile göstermektedir.Hücre boyutunun ayrışması d = / 6. İçeren kanal Şekil l'deki kesikli yol, koyu renkli hücrelerle gösterilir.Ana hatlarını. Böyle bir kanal,Ancak, birbirini takip eden hiçbir çift hücrenin tek başına olanaksız, yani iki ardışık tarafından tanımlanan bir kanal tek başına hücreler her zaman uygulanabilir bir yol içerir. Başka bir deyişle, iki bitişik hücre için hücreye bağımlı özellik yoktur.Cezalandırılabilen iki bitişik hücre ile ilişkili grafik aramayı önlemek için bir kenar maliyet fonksiyonu ile gösterilir.

Ayrıca uygulanabilir bir yolun garanti edildiği ileri sürülebilir.Hücrelerin boyutları büyükse herhangi bir kanalda var olmak yeter. Gerçekten, Ref. [17] eğrilik sınırının 1 / rmin değerine eşit veya daha düşük yerel eğriliğe sahip yol kanalın genişliği w,w ality τrmin eşitsizliği, burada τ belirli bir polinomu tatmin eder denklem. Yukarıdaki karşı örnek aynı zamanda böyle bir hücre seçiminin pratikte çok kısıtlayıcı olabileceğini. hücrelerin boyutları ortamın ayrıntılarını yakalamak için çok büyük olabilir.

**B- Birden Fazla Hücreyle Çalışma**

Bu yazıda, plana aşağıdaki yaklaşımı öneriyoruz yolu birleştirirken hücre ayrışmaları kullanan bir yol eğrilik bilgileri: İlişkili topolojik grafikte belirli bir hücre ayrışması ile bir maliyet fonksiyonu tanımlarız.Bazı sabit k> 2 için, düğümlerin k − tupleslerine dayanarak,her bir k-demetinin elemanlarının çift bitişik olduğu.K-tuples yoluyla çapraz geçişin fizibilite sorunu(birbirini izleyen iki hücre arasında geçiş yapmak yerine)sadece maliyet fonksiyonlarının daha genel tanımlarına izin verir. İçinde özellikle,içeride yatan herhangi bir yol için maksimum yaklaşık eğrilik Kanal. Sonuç olarak, mümkün olan yörünge her zaman hesaplanan kanalın içinde bulunur daha önce bile geometrik, yol planlama katmanındaki hücrelerin hareket planlama görevini çağırmak. bu not alınmalı benzer fikirler araştırılmıştır.

**Algoritma**

Belirli Bir Başlangıç ​​Zamanı İçin En Kısa Yol Algoritması. Bir kaynak verildi düğümler ve başlangıç ​​zamanı ts, en kısa yolları ve minimum gecikmeleri ararız Bu başlangıç ​​zamanı için s ve diğer tüm düğümler arasında. Dreyfus'te belirtildiği gibi, bu Dijkstra’nın veya Ford’unun algoritmalarının basit bir uzantısıdır. WA bazı kavramlara ve farklılıklara aşina olmak için algoritmayı burada sunun daha sonra yardımcı olacak standart sürümden. Şimdi bir etiketlemenin sürümü olan algoritmayı sunmaya devam ediyoruz her düğümün mümkün olan en erken varış zamanı ile etiketlendiği algoritma kaynak düğümde verilen başlangıç ​​zamanı olan düğüm. Terminolojisini takip etmek etiketleme algoritmaları, ve düğümün kalıcı etiketidir (NULL,düğüm kalıcı olarak etiketlenmez) ve Yk geçici etiketidir. Algoritma ayrıca tarafından oluşturulabilen en kısa topolojik yolları oluşturur.Algoritma tarafından hesaplanan değerler. fk buradaki k düğümünün babasının kimliğidir ağacı. Yukarıda tanımlandığı gibi Dik (t) fonksiyonları algoritma girdisinin bir parçasıdır. Yaptığımız algoritma mevcut UW 1'in genelleştirilmesidir. Temel zorluk,X değerleri artık zamanın işlevleridir. Ayrıca, yayılmayı oluşturma Ağaç zamanla değiştiği için ağaç artık daha karmaşık yakalamak daha zor. Bu ikinci sorunun üstesinden gelmek için, tek bir değer yerine,her bir k düğümü ve komşularının her biri için tanımlanan Y ,, (t) fonksiyonları 1 E Nk. & (t), daha önce olduğu gibi, başlangıç ​​noktası t için k düğümündeki en erken varış zamanıdır.(belirli bir yürütme adımında bilindiği gibi). Ykl (t) benzer şekilde en erken varış komşu düğüm k üzerinden, t zamanında başlayan bir mesajın 1. düğümünde.Yayılan ağacı belirlemek için Y üzerindeki çift indeks kullanılır. Bu ağaç tarafından tanımlanır düğüm 1 ve zaman için bir baba k'nin seçilmesi, böylece Xt (to) = Yk [(to); daha fazlaysa böyle bir düğüm var, en küçük endekse sahip olanı (keyfi olarak) seçiyoruz (bu, seçim benzersiz olmadığında kağıt boyunca yapılır).

**Sezgisel Dinamik Programlama**

Bu alt bölümde, DH olarak adlandırılan dinamik bir programlama tabanlı buluşsal algoritma sunuyoruz. İlk önce elektrikle ilişkili bir dizi durum tanımlarız ve düğümlerdeki benzin seviyeleri. Sonra Bellman denklemlerini sunuyoruz (Bellman 1956)Bunu kolaylaştırmak için minimum maliyet yolu uzunlukları tarafından karşılanması gerekir.Dinamik programlama çözümü metodolojisi. Son olarak, bu denklemlerin çözümünün verimli bir şekilde gerçekleştirilebileceği bir grafik dönüşümü sunuyoruz dönüştürülmüş grafik üzerinde en kısa yol problemini çözerek.Bir düğüme gelişi temsil eden bir hi, σ, λi üçlüsüdür σ ∈ [P, P] kWh elektrik yükü ve λ ∈ [G, G] galon benzinli i ∈ N.Notasyonu kullanacağız ωσ, λ ben bir duruma başvurmak ve bu gösterimi ω ile değiştirmek için veya ωi, bağlamın ayırt edilmesi için belirli i, σ ve λ değerlerini gerektirmediğinde.Bir E-MCPP-PHEV örneği hV, X, s, t, Ps, Gs, Pt verildi, Gti, bir çözelti hx, e +, g + i x 'den x vektöründen çıkarılabilen s'den t'ye bir yol içerir. İle özgül enerji (e+) ve benzin (g+) düğümlerden alımlar, mesafeler CD ve CS modlarında bu yolla kolayca çıkarılabilir. Birlikte Ps ve Gs ile x, e + ve g vektörleri + şarj durumu ile ilgili çözüm yolundaki düğümlere varışta benzin. Yani, her çözüm için E-MCPP-PHEV, bu çözümü temsil eden benzersiz bir durum dizisi vardır.Genel olarak, E-MCPP-PHEV'in sayılmaz sayıda uygulanabilir olduğuna dikkat edin çözümleri. Bu çözümlerin her biri benzersiz bir durum dizisiyle eşleştiğinden,devlet alanı da sayılamaz. Ancak bu sayılamayan durum alanı DH'nin arkasındaki ana fikir olan sonlu bir yaklaşımla.Space, τ ∈ N durum uzayı için ayrıklaştırma parametreleri olsun. Düşünmek iki set Σ = {σ0, σ1,. . . , σξ} ve Λ = {λ0, λ1,. . . , λτ} burada σ0 = P, λ0 = G,σk = σ0 + k × P –P ξ ∀k ∈ {1, 2,. . . , ξ} ve λl = λ0 + l × G-G, τ ∀l ∈ {1, 2,. . . , τ}.

**Sonuçlar**

Makale, multipleorigin-multip-destination (MOMD) sorunu adı verilen bir yol bulma problemi önermekte olup, burada amaç, kullanılan yayların toplam maliyetini en aza indirmektir. Bu sorun, ulaşım ve ağ sorunları nedeniyle motive oluyor burada yollar arasındaki varyasyon en aza indirilmelidir.Biz bu sorunun NP-tam olduğunu gösterdi ve Sorunu en iyi şekilde çözmek için SAT tabanlı modeller. Model akışı koruyan kısıtlamalara dayanarak hesaplamalı görünüyor ulaşılabilirlik kısıtlamalarına dayalı modelden daha verimlidir. İlk deneyler ayrıca SAT modeli CP ve ASP modellerinden daha verimlidir.Çözülmesi gereken birkaç açık sorun var. İlk olarak,her iki SAT modelini, özellikle Flow modelindeki karar değişkenlerinin sayısı nedeniyle, farklı sayıda paketle ilgili problemleri kullanarak karşılaştırmak ilginç olacaktır.Ulaşılabilirlik modelindeki değişken sayısı sayıdan bağımsız iken paket sayısına bağlıdır paketler. İkincisi, daha iyi çalışmak ilginç olurdu nesnel işlev için daha düşük sınırlar, özellikle daha iyi sezgisel hesaplamanın orijinal motivasyonu ile ilgili olarak Ulaştırma problemi için tahminler. Üçüncüsü, SAT çözücülerin verimliliğini son teknoloji MIP çözücülerle, özellikle de aşağıdakileri kullanan Flow modeli için karşılaştırmak ilginç olacaktır. En kısa bulmak için bir algoritma sunduk arasında geçiş yaparken genel grafiklerde mesafe yolları grafik düğümleri önceki yol geçmişine bağlıdır. Önerilen algoritması Dijkstra’nın algoritmasının bir uzantısını temel alır iki düğümlü geçişler için. Bu algoritmayı yol planlama ve hareket planlama probleminin çözümü kinodinamik kısıtlamalara tabidir. Eğrilik dahil ederek yol (geometrik) planlama adımında uygulanabilir bir yol hakkında bilgi, hareket planlamasında daha iyi yollar sağlayabiliriz iki planlama adımı,tipik olarak mevcut uygulama gibi sırayla uygulanır.

**Program Kodları**

7.Satırdaki kod projenin ana başlığı gibidir içinde değer adlı bir iki boyutlu dizi tanım oluşturulmuştur.10.Satırdaki kod maksimum adlı bir ifade oluşturup değer adlı tanımın uzunluğuna eşitliyoruz.11.Satırdaki kod minimum adlı bir ifade oluşturup değer adlı tanımı “0” alarak uzunluğuna eşitliyoruz.14.Satırdaki kod t adlı bir iki boyutlu dizi belirliyoruz ve bunu maksimum ve minimum adlı değerlere eşitliyoruz.16.Satırdaki kod bir döngü oluşturuyoruz bu döngünün içinde i adlı değer oluşturup “0” a eşitliyoruz sonra maksimim değer i değerinden küçüktür şeklinde tanımlıyoruz.ardından i değerini “1” arttır diyoruz.18.Satırdaki kod bir döngü oluşturuyoruz bu döngüde j adlı bir değişken oluşturup “0” a eşitliyoruz.sonrasında minimum değer j ifadesinden küçük olarak yazıyoruz.ardından j ifadesini “1” artırıyoruz.20.Satırdaki kod T adında bir iki boyutlu dizinin içine j ve i elemanlarını yazıp değer adlı iki boyutlu dizimizin içine de i ve j ifadelerini yazıp eşitliyoruz.23.Satırdaki kod eğer i ifadesi 0 eşit ve j ifadesi “0” dan büyük ise olarak belirtiyoruz.24.Satırdaki kodda T dizisinin içine 0 değerini ve j ifadesini yazıyoruz ardından T dizisinin içine 0 değerini ve j ifadesinden “1” çıkartıp eşitliyoruz ardından bu ifadeyi t dizisine ekliyoruz.28.Satırdaki kodda değil ise olarak belirtip j ifadesini 0 eşitliyoruz ve i ifadesi “0”dan büyüktür olarak tanımlıyoruz.29.Satırdaki kod T dizisinin içine i ifadesini ve aynı zamanda “0” değerini yazıyoruz ardından T dizisinin içine i ifadesinden “1”çıkartıyoruz ve “0” değerini yazıyoruz.bu ifadeye eşitleyip topluyoruz.33.Satırdaki kod değil ise olarak belirtip ifadesi “0” dan büyük ve j ifadesi “0” dan büyük olarak belirtiyoruz.34.Satırdaki kod T dizisinin içine i ifadesini ve j ifadesini tanımlayıp bulduğumuz en düşük sayı içerisindeki T dizisinin içerisine de i ifadesinden “1” çıkarıyoruz sonra j ifadesini ekliyoruz,T dizisine i ifadesini ve j ifadesinden “1” çıkartıp ekliyoruz bu yazdığımız ifadeleri de T dizisinin içine i ifadesi ve j ifadesi belirlediğimiz diziye ekliyoruz.38.Satırdaki kod ise T dizisinin içine maksimum değerden “1” çıkartıp ekliyoruz ve minimum değerden “1”çıkartıp ekliyoruz ve bu ifadeye geri dön olarak belirtiyoruz.39.Satırdaki kod main ifadesi açıyoruz.40.Satırdaki kodda matris adlı iki boyutlu dizi belirliyoruz .42.Satırdan 46.Satıda kadara olan yerlerde matrisimizi belirliyoruz 5 değer açısından belirliyoruz bu değerler(A,B,C,D,E).48.Satıdaki kodda belirlediğimiz matrisden en az maliyetli ve en kısa yolu ekrana yazdırıyoruz.

**PROJE BİLGİLERİ**

**Proje Konusu**

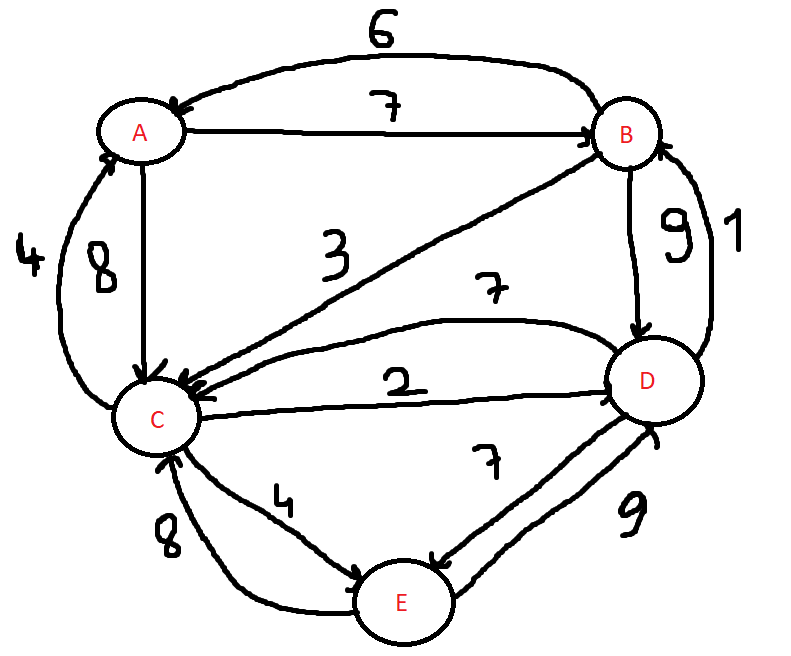
Proje konusu herhangi bir bireyin yada matematiksel hesaplama olarak olsun,veya bir programa bunu yaptırmak amaçlı olsun,verilen projenin nasıl en kısa sürede ve en iyi halde teslim olacak konularında yardımcı olan bir projedir.Projemde minimum ve maksimum yollarını belirleyerek bir başlangıç ve varış noktasından en kısa yani minimum yolu buluyor.Matrisi kendimiz hangi yol hangi yola kadar kaç değer içeriyor ise onları yazıp en ucuz maliyetli ve kısa yolu buluyor.

**Algoritma Yapısı**

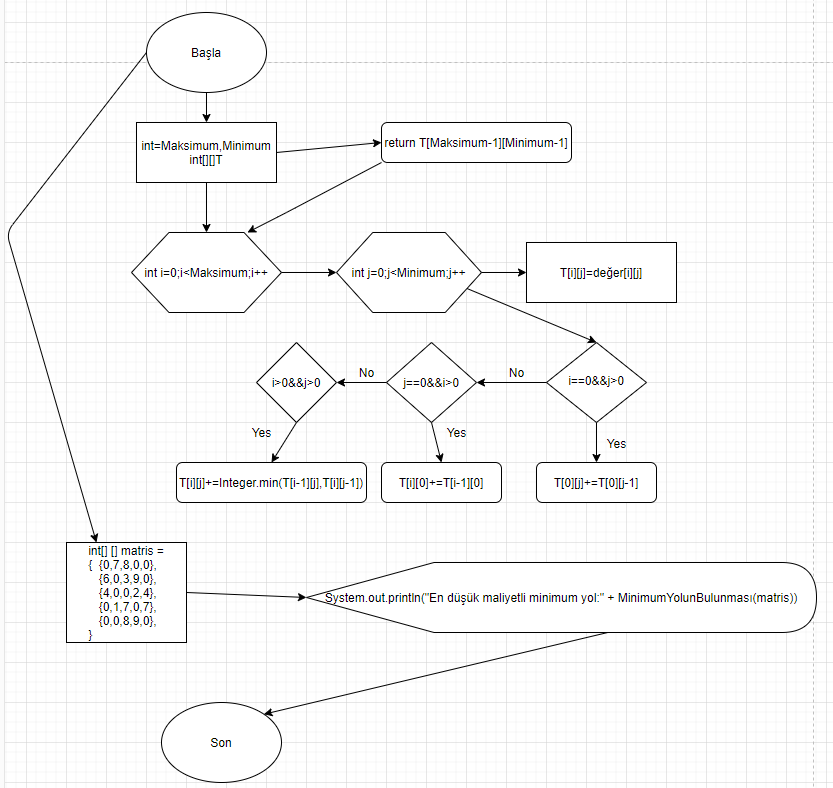
Başlangıç noktası(0,0) dan başlar.Tanımladığımız dizinin içine belirlediğimiz minimum ve maksimim yolu diziye belirtiyoruz.Herhangi bir noktadan başlıyacağımız anlamına geliyor. Bir tane for döngüsü oluşturarak ilk önce maksimum yolun ve ardından bir tane daha for açarak minimum yolu yazıyoruz.Nereden nereye gideceğimizi belirleyip ardından hangi yollardan gitme ihtimalimiz var onları belirliyoruz. Minimum yolun içine Belirlediğimiz “T” dizisinin içine maksimum ve minimum yollardan küçük olarak belirlediğimiz ifadeleri yazıyoruz.Burada bizim başlangıç noktamıza en yakın yeri belirliyoruz. Minimum yoldan küçük olan ifade “0”’a eşit ise ve aynı zamanda maksimum yoldan küçük olan ifade “0”’dan büyük ise “T” isimli oluşturduğumuz diziye maksimum yoldan küçük olan ifadeden “1” çıkartım dizimize ekliyoruz.Bize en yakın yer olarak belirlediğimiz yolda eğer belirlediğimiz değerden büyük hesaplanırsa tekrardan hesaplama işlemini gerçekleştiriyoruz.Eğer maksimum yoldan küçük ifade “0” a eşit aynı zamanda minimum yoldan küçük ifade “0”’dan büyük ise “T” dizisindeki elemanlardan minimum yoldan küçük ifadeden “1” çıkartılarak eklensin.Bize en uzak olan noktayı belirliyoruz ve ardından bu yol belirlediğimiz en uzak yol değerinden fazla ise tekrar hesaplamak gerekiyor. Matrise ulaşmak,bitişik olan hücreleleri,sol hücre veya bitişik üst hücreden göstermek için değil ise ile başlıyoruz.Gideceğimiz hedefi tekrardan gidebileceğimiz bir konumdamı onu hesaplıyoruz.Maksimum yoldan küçük ifade “0”’dan büyük ise ve aynı zamanda minimum yoldan küçük olan ifade “0”’dan büyük ise t dizimizden minimum ve maksimum yoldan küçük ifadelerden bir çıkarıp dizimize ekliyoruz.Gideceğimiz noktaya en uzak ve en yakın yollar değerleri hesapladığımız gibi çıkmaz ise bu hesapladığımız yolların değerlerini gideceğimiz yolun değerlerinden çıkarıp tekrar başlangıç noktasından gideceğimiz noktaya olan hesabımızı yapıyoruz”T” hücresinin hedefe ulaşması için minimum maliyetini koruması için minimum ve maksimum yoldan “1”çıkarılarak geri dönmesini sağlıyoruz.Ulaşacağımız noktaya en kısa yoldan herhangi bir yol daha ekliyerek daha da kısaltmayı amaçlıyoruz.Asıl main sınıfmızda bir maliyet sınıfı oluşturarak kendimiz bir matris belirliyoruz.Gideceğimiz noktaya olan bütün yolların değerlerini belirleyerek grafiğini çıkarıyoruz. Bu belirlediğimiz matris sınıfa göre program minimum maliyetli ve en kısa yolu bize söylemiş oluyor.Ve son olarak bu grafikten hangi yolun bize daha kısa sürede hedefimize götürdüğünü bulup o yoldan yani minimum yoldan hedefimize ulaşmış oluyoruz.

**Minimum Yolun Bulunması Uygulaması Tasarımı**

**Graf Yapısı**



**Akış Şeması**



**ÖZ GEÇMİŞ**

En düşük maliyetli çalışmalar için CBS yazılımını uygulayan birçok arkeolog tam olarak en düşük maliyetli ağlar oluşturmak için kullanılan fonksiyonları anlamak ve böylece yazılım tarafından uygulanan modelin önkoşulları ve dezavantajları. Bu kağıt Harita üzerinde nokta bağlamak için en düşük maliyetli ağlarla sonuçlanan birkaç modeli tartışır: en kısa yolları eşleştirin, inşaatçıya en düşük maliyetli ağı, en yakın ağa bağlanın komşular, nirengi ağı, minimum yayılan ağaç, merkezi nokta ağı ve ana yardımcı yollar ile seyahat rotası. Modellerin çoğu,Bergisches Land, Almanya. Bu bölge doğal ortamlardan daha az çekici doğal koşullar sağlar.Ren'e yakın bir alandadır ve bu nedenle bu bölgede çok az yerleşim vardır.MS 1000 yılından önce. Detaylı tarihsel kaynaklar ve eski haritalar kontrol edilebilir modellerin tarihsel gerçekliğe uygun olup olmadığı.

**GİT HUB LİNK**

[**https://github.com/geggs123/18MY03015-EMIRHAN-KARAPINAR-MINIMUM-YOLUN-BULUNMASI-PROBLEM-PROJESI**](https://github.com/geggs123/18MY03015-EMIRHAN-KARAPINAR-MINIMUM-YOLUN-BULUNMASI-PROBLEM-PROJESI)

**KAYNAKÇA**

[**http://www.nuhazginoglu.com/tag/en-kisa-yol-problemi/**](http://www.nuhazginoglu.com/tag/en-kisa-yol-problemi/)

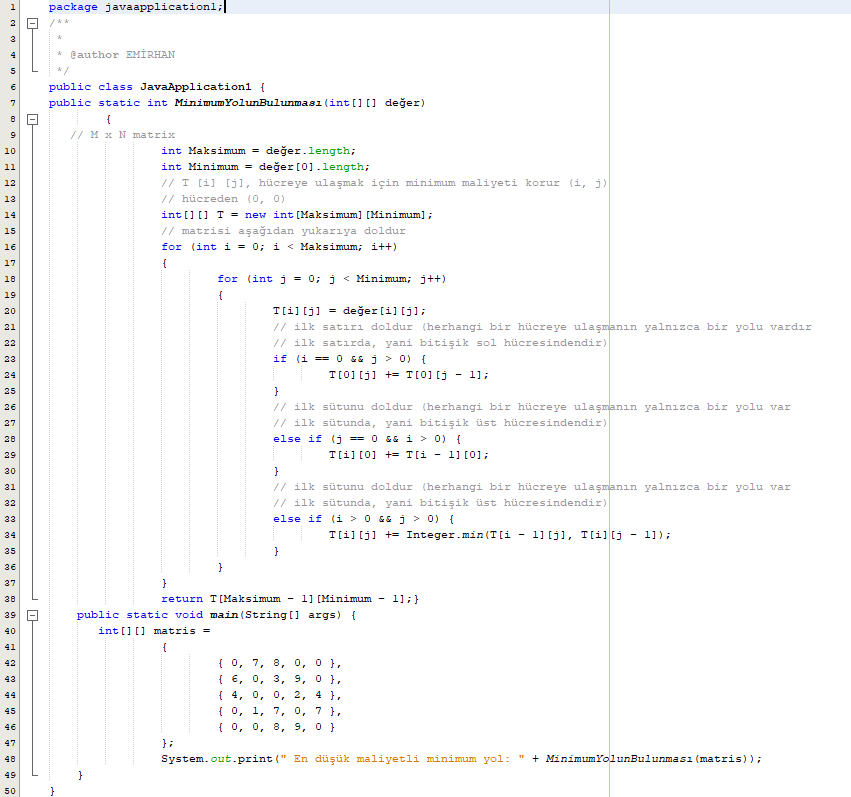
[**http://bilgisayarkavramlari.sadievrenseker.com/tag/minimum-spanning-tree/**](http://bilgisayarkavramlari.sadievrenseker.com/tag/minimum-spanning-tree/)

[**https://www.cancankiran.com/eslesme-algoritmasi-graf-kurami/**](https://www.cancankiran.com/eslesme-algoritmasi-graf-kurami/)

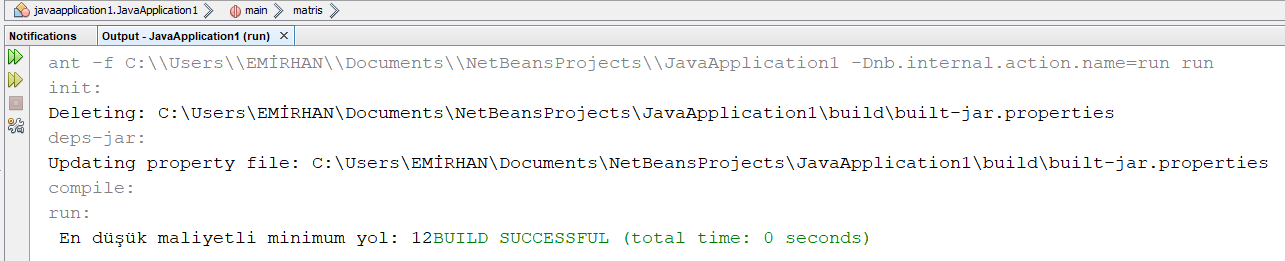
[**http://abl.gtu.edu.tr/hebe/AblDrive/59669005/w/Storage/104\_2011\_1\_511\_59669005/Downloads/ayrik-ilerikonu-bol6-9.pdf**](http://abl.gtu.edu.tr/hebe/AblDrive/59669005/w/Storage/104_2011_1_511_59669005/Downloads/ayrik-ilerikonu-bol6-9.pdf)

**EK**

**Proje Kodları**



**Projenin Ekran Çıktı Görüntüsü**



**Projemi Nasıl Yaptım**

File new Project e tıklayıp Java with ant dosyasını seçip javaApplication a tıklayarak yeni bir proje oluşturdum.Önüme bir class ve main kodlarımızı yazıcağımız yer geldi.classımın içine bir MinimumYolun bulunması adında kodlarımın işleneceği yeri oluşturdum bu yeri oluştururkende değer adlı iki boyutlu dizi tanımladım.Bu yerde kendime 2 değişken tanımladım minimum ve maksimum yol olarak bu yolları tanımladığım değer adlı ifadelerin uzunluklarına eşitledim tabi ki minimum ve maksimum değerlerin farklı özellikler vererek.Kendime bir tane daha T adlı iki boyutlu bir dizi oluşturdum ve bunların içine maksimum ve minimum değerleri belirledim daha sonra bir döngü oluşturdum minimum değer döngüsü ve ardından içine maksimum değer döngüsü 2 döngü oluşturdum bu en sonuncu döngünün içine T adlı dizimizi değer dizimize eşitledim ve özelliklerini yazdım.Ardından if olarak bir oluşturdum ve içlerine ifadeleri yazdım if in içine de T dizisine eklenecek T dizisindeki ifadelerin özelliklerini vererek ekledim.else if oluşturudm ve özelliklerini yazdım ardından bunun içine T dizisinin içine kendi T dizisine özellik vererek kendisine tekrardan ekledim.tekrardan else if oluşturup özellikleri yazıp içine T dizisini en düşük maliyetini ve en düşük yolu bulması için özellik vererek kendisine ekledim.for döngülerini kapatıp en son da maksimum ve minimum değerlere tekrar dönmesini sağladım.Main kodlarımı yazıcağım yere matris adlı iki boyutlu dizi belirledim sonra içine matris belirledim bu matris 5 değerli bir matris.En sondada en düşük maliyetli minimum yolu ekrana yazdırmasını istedim.