

# Gezgin Satıcı Problemi

Haldun Sural\* / [sural@ie.metu.edu.tr](mailto:sural@ie.metu.edu.tr)



Gezgin Satıcı Problemi'nde (GSP) amaç, bir satıcının, bulunduğu şehirden başlayıp, her şehre sadece bir kez uğradıktan sonra başladığı şehre geri dönen en kısa turu bulmaktır. Herhangi iki şehir arasında bir yol olduğunu ve o yolun uzunluğunu bildiğimizi varsayıyoruz.

Görüldüğü gibi, GSP, anlaşılması için matematiksel herhangi bir temel gerektirmeyen bir problemdir. Anlaşılması kolaydır ama çözümü zordur!

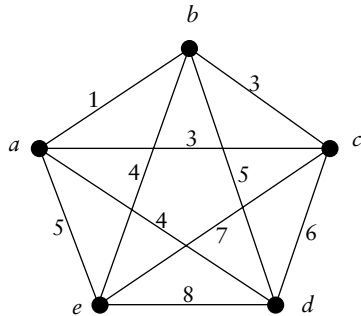


Hamilton

GSP, çizge kuramı dilinde, şehirlerin noktalarla, şehirlerarası yolların kenarlarla temsil edildiği (yaln) bir çizge üzerinde, en kısa Hamilton turunun bulunmasıdır. **Hamilton turu**, bir çizge üzerindeki her noktadan sadece bir kez geçen (dolayısıyla aynı yoldan da sadece bir kez geçen) ve başladığı noktada biten, 19. yüzyılda yaşamış matematikçi William Hamilton'ın adıyla anılan turdur.

Örneğin  $n$  noktadan oluşan bir tamçizge, yani  $K_n$  tamçizgesi  $(n-1)!/2$  Hamilton turu içerir. Bunu Tam Tur Olasılığı adlı yazıda kanıtlamıştık (sf. 32).

**Soru 1.** Yan-daki çizgede bütün Hamilton turlarını bulun ve uzunluklarını hesaplayın. Şehirlerarası uzaklıklar kenarların üstünde verilmiştir.



**Yanıt 1.** Verilen çizgede her nokta çifti arasında bir kenar bulunduğu için, bunun bir tamçizge olduğunu belirtelim. Bir tamçizgede, noktaların herhan-

gi bir sırayla dizilişi, bir Hamilton turuna karşılık gelir. Örneğin,  $a$  şehrini başlangıç noktası kabul edersek, aşağıda verilen  $(5-1)!/2 = 12$  turu buluruz.

Tur	Uzunluk
$abcdea$	$1+3+6+8+5=23$
$abceda$	$1+3+7+8+4=23$
$abdcea$	$1+5+6+7+5=24$
$abdeca$	$1+5+8+7+3=24$
$abecda$	$1+4+7+6+4=22$
$abedca$	$1+4+8+6+3=22$
$acbdea$	$3+3+5+8+5=24$
$acbeda$	$3+3+4+8+4=22$
$acdbea$	$3+6+5+4+5=23$
$acebda$	$3+7+4+5+4=23$
$adbcea$	$4+5+3+7+5=24$
$adcbea$	$4+6+3+4+5=22$

Burada en kısa tur için 22 birim uzunluğunda dört seçenek vardır. Bu dört turdan herhangi birini, örneğin  $abecda$  turunu, bu GSP'nin eniyi çözümü olarak kabul edebiliriz.

Bu örnekteki çözüm yöntemini izleyerek, GSP için üç adımlık bir çözüm yolu geliştirilebilir:

1. Çizgenin tüm Hamilton turlarını bul.
2. Her turun uzunluğunu hesapla.
3. Turlar arasından en kisasını seç.

Bu çözüm yöntemiyle, 10 şehir içeren bir GSP için bulunması gereken tur sayısı  $9!/2 = 181.440$ 'tır. Şehir sayısı 20'ye çıktığında ise bulunması gereken tur sayısı  $19!/2 \approx 6,08 \times 10^{16}$ 'yı bulur. 25 şehir için GSP problemini bu yolla çözmek isteyen bir satıcının, yaklaşık  $3,1 \times 10^{23}$  turu incelemesi gerekir. Eğer satıcı, 25 şehirli bir GSP problemini, her Hamilton turunu  $10^{-9}$  saniyede inceleme kapasitesine sahip bir bilgisayarla çözmeye kalkarsa, ancak 10 milyon yıl sonra en kısa turu bulabilir... Bulunan çözüm çözüm olmasına çözümün de, çözüm yolunun uygulanması olanaksız.

**Algoritmalar ve Çözüm Karmaşıklığı.** Algoritmalar çözüm karmaşıklığı açısından genel olarak iki grupta incelenir. Eğer çözüme ulaşmak için algoritmanın yapacağı işlem sayısı, problemin boyutunu belirleyen verilerin bir polinomu olarak ifade edilebiliyorsa, bu algoritma genellikle "etkin",

\* ODTÜ, Endüstri Mühendisliği Bölümü öğretim üyesi.

“iyi” veya “çabuk” bir algoritma olarak kabul edilir. Örneğin  $n$  ve  $m$  doğal sayılarıyla ilgili bir algoritma  $n^2m^3$  (ya da  $n^5 + nm$ , yani  $n$  ve  $m$ 'nin çeşitli çarpımlarının toplamı) işlemden sonra sonuca ulaşıyorsa, bu algoritma hoşumuza gider, algoritmayı “çabuk” olarak nitelendiririz. Ama eğer işlem sayısı, verilerin üssel kuvvetiyle ifade edilen bir algoritmaysa, bu, “kötü” bir algoritmadır. Örneğin  $n$  doğal sayısı ile ilgili bir algoritma  $2^n$  işlem sonra sonuca ulaşıyorsa, bu algoritmayı pek beğenmeyiz, çünkü  $2^n$ ,  $n$ 'ye göre çok çok büyük bir sayıdır;  $2n$  için yapılan işlem  $n$  için yapılan işlemin karesi kadar artar.

Algoritmaların “iyi” ve “kötü” olarak gruplandırılması gibi, problemleri de bilinen çözümlerinin karmaşıklığına göre “kolay” ve “zor” olarak sınıflandırabiliriz: Polinom zamanlı bir algoritmayla çözülebilen problemler “kolay” problemler sınıfını, böyle bir çözüm yöntemi bulunamamış problemler ise “zor” problemler sınıfını oluşturur. GSP zor problemler sınıfına ait bir problemdir.

Eğer GSP'nin zor bir problem olduğuna ikna olmadıysanız aşağıdaki soruyu çözün (daha doğru- su çözmeye çalışın!)

**Soru 2.** *Bir Türkiye karayolları haritasında verilen illerarası mesafe bilgilerini kullanarak, bulunduğunuz şehirden başlayıp yine aynı şehirde biten ve 81 ili dolaşan bir tur belirleyin. Bulduğunuz tur, bulunabilecek en kısa tur mu?*

**GSP Çözüm Yöntemleri.** Araştırmacılar, GSP'nin kökenini Euler'in 1759 ve Vardemonde'nin 1771'teki çalışmalarına dayandırır. Ancak, 1940'ların sonlarına doğru popüler olmaya başlayan GSP için 1954'te Dantzig, Fulkerson ve Johnson, doğrusal programlama tekniğine dayalı bir çözüm yöntemini geliştirmiştir. GSP için geliştirilen çözüm yöntemlerinin “ilk”i kabul edilen bu yaklaşım, “polyhedral combinatorics” alanının gelişmesine de öncülük etmiştir. Ancak, GSP'nin zor problemler sınıfına ait olmasından da anlaşılacağı gibi, bu yaklaşım dahil, bugüne kadar geliştirilen en iyi algoritmaların bile çözüm karmaşıklığı, problemdeki şehir sayısını temsil eden  $n$ 'nin üssel kuvvetine bağlıdır. Büyük bir olasılıkla polinom zamanlı bir GSP algoritması da hiçbir zaman bulunamayacaktır, hatta büyük bir olasılıkla böyle bir algoritma yoktur.

Çok şehirli GSP'leri çözmek için pratik bir yöntem, yaklaşık çözüm üreten sezgisel algoritmalar kullanmaktır. Sezgisel algoritmalar, eniyi çözümü garanti etmemelerine karşın, eniyi çözüme yakın, oldukça iyi bir sonucun makul bir sürede bulunmasını sağlarlar. Şimdi bu tür çözüm yaklaşımlarına iki örnek verelim.

### En Yakın Şehir Algoritması

Bu sezgisel algoritma, gezginin bir şehirden diğerine giderken, henüz ziyaret etmediği şehirler arasından en yakındakini tercih edeceği ilkesine dayanır.

**Soru 3.** *Birinci soruda verilen çizge için  $a$  şehrini başlangıç noktası kabul ederek “en yakın şehir algoritması”yla bir çözüm bulunuz.*

**Yanıt 3.** Satıcı  $a$ 'dayken henüz ziyaret edilmemiş şehirler kümesi  $\{b, c, d, e\}$ 'dir. Bu küme içinde  $a$ 'ya en yakın  $b$ 'dir ve algoritmaya göre satıcı önce  $b$ 'ye gider.  $b$ 'ye en yakın ziyaret edilmemiş şehir  $c$ 'dir. Dolayısıyla satıcı  $c$ 'den  $d$ 'ye geçer. Buradan  $e$ 'ye gider, sonra da başlangıç şehrine geri dönerek turunu tamamlar. Buna göre 23 birim uzunluğundaki  $abcdea$  turu bulunur. Ancak, bu çözüm maalesef çizgedeki en kısa turdan bir birim daha uzundur (bkz. Soru 1).

Eğer bir çizgede tüm  $i, j, k$  noktalarının (şehirlerinin) arasındaki uzaklıklar  $d_{ij} \leq d_{ik} + d_{kj}$  eşitsizliğini sağlıyorsa, bu çizge “üçgen eşitsizliğini sağlayan çizge” olarak adlandırılır. Bunun yararı, böyle bir çizgede iki nokta arasındaki en kısa yolun, o iki nokta arasındaki tek kenarla sağlanacağını garanti edilmesidir. Çizgemizin noktaları şehirler, noktalar arasındaki uzaklık da şehirlerarası en kısa mesafe ise, bu varsayım geçerlidir elbette.

**Soru 4.** *İlk soruda verilen çizgenin “üçgen eşitsizliği”ni sağladığını kanıtlayın.*

Şimdi üçgen eşitsizliğini sağlayan bir çizgede, en kısa Hamilton turunun en fazla iki katı uzunluğunda bir Hamilton turu bulan sezgisel bir algoritmayı inceleyelim.

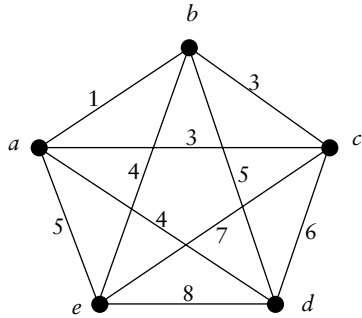
**En Kısa Tur Algoritması.** Bir  $G$  çizgesi üzerindeki en kısa kapsayıcı ağacı  $T$ 'yle gösterelim. Yani  $G$ 'nin bütün noktaları  $T$  ağacında bulunsun ve

$T$ 'nin kenarlarının uzunluklarının toplamı olabilirince küçük olsun.

$T$ 'nin her bir kenarı için ikinci bir kenar ekleyerek, yeni bir  $G_1$  çizgesi oluşturalım. Bu yeni çizge yalın bir çizge olmasa da gene de bir çizgedir.  $G_1$  çizgesinde bütün noktalar birbirlerine çift sayıda kenarla bağlanmış olur, yani her noktanın derecesi çifttir. Böyle bir çizgede, Euler turu olarak adlandırılan, bütün kenarlardan sadece bir kez geçen, dolayısıyla her noktadan en az bir kez geçen ve başlama noktasına geri dönen bir tur vardır, bunu Euler Turu yazısında görmüştük.  $G_1$  çizgesinde bulduğumuz bir Euler turunun noktalarını rastladığımız sıraya göre dizelim. Bu turu bir Hamilton turuna dönüştürmek için, sıraya dizilmiş noktalar arasında tekrar edilmiş bir noktaya rastladığımızda, bu noktayı atlayıp bütün noktaların turda sadece bir kez yer almasını sağlayarak, başlama noktasında tamamlanan yeni bir tur bulmamız yeterlidir. Eğer çizge üçgen eşitsizliğini sağlıyorsa, tekrar eden noktaların atlanmasının turun uzunluğunu değiştirmeyeceği hatta belki de kısaltacağı muhakkaktır.

Şimdi ilk soruda verilen çizge üzerinde bu algoritmayı uygulayalım.

Önce, Prim algoritmasıyla çizge üzerinde en kısa kapsayıcı ağacı, yani çizgenin her noktasını içeren en kısa ağacı bulalım. Bu algoritmada, önce, çizgenin kenarları en kısıdan en uzuna doğru sıraya dizilir. Soruda verilen çizge için bu sıra şöyle oluşur:  $ab, bc, ac, be, ad, ae, bd, cd, ec, ed$ . Sıranın en başındaki en kısa kenarla başlarız. Bu kenarın iki noktası, yani  $\{a,b\}$  noktaları en kısa kapsayıcı ağacımızın iki noktası olacak. Sonra, soldan sağa doğru kenarları gözden geçiririz. O ana dek seçtiğimiz noktalara kattığımızda ağaç özelliğini bozmayacak ilk kenarı seçtiğimiz noktalara ekleriz. Sıradaki ikinci en kısa kenar, ya bu ağaca yeni bir nokta ekler ya da iki kenar ve dört noktadan oluşan bir orman verir. Ağaç elde edersek bu kenarı da katarız, yoksa bir sonraki noktaya geçeriz. Örnek çizgede  $bc$  kenarı ağaç özelliğini bozmaz, dolayısıyla  $c$ 'yi de katarak  $\{a,b,c\}$

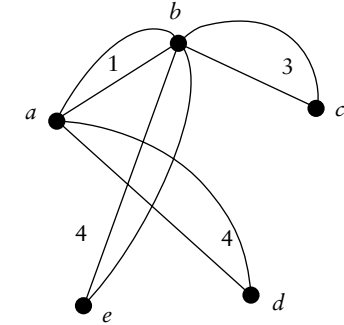
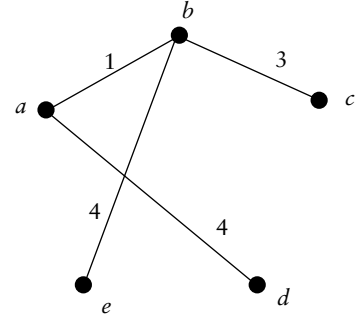


noktalarından oluşan bir ağaç elde ederiz. Üçüncü  $ac$  kenarı ağaç özelliğini bozuyor, dolayısıyla bu kenarı atlayarak, dördüncü kenara geçmeliyiz. Bu şekilde devam edersek, çizgedeki bütün noktaları içeren en kısa kapsayıcı ağacı buluruz.

Sağdaki  $T$  ağacı,  $G$  çizgesinin en kısa kapsayıcı ağacıdır. Uzunluğu  $1 + 3 + 4 + 4 = 12$ 'dir. Çizgede daha kısa bir kapsayıcı ağaç bulunamaz. Şimdi  $T$ 'deki kenarları çiftleyelim ve oluşturduğumuz yeni çizgeye  $G'$  diyelim.

Daha önce söylediğimiz gibi, eğer bir çizgenin noktaları çift sayıda kenarla bağlarsa, Euler turunu bulmak kolaydır ve nasıl bulunacağı Euler Turu yazısında açıklanmıştır. Örneğin, bu çizge üzerinde  $ebc-badabe$  bir Euler turudur. (Çiftlenen kenarlar arasında bir fark gözetmedik.)

$e$ 'den yola çıkan bir gezgin,  $eb$  kenarından  $b$ 'ye,  $bc$  kenarıyla  $c$ 'ye,  $cb$  kenarıyla tekrar  $b$ 'ye ulaşır. Buradan da  $ba$  kenarıyla  $a$ 'ya ulaşan ve  $a$  noktasından itibaren geriye kalan bütün kenarların üzerinden sırayla geçen gezgin, en son olarak tekrar  $e$  noktasına erişir. Euler turunu izleyen gezgin, örneğin, turun başlarında  $c$ 'den tekrar  $b$ 'ye döner. Şimdi, bu turu Hamilton turuna dönüştürürken, bu aşamada ikinci kez ziyaret edilen  $b$  noktasının ikinci ziyaretini engellememiz gerekir. Yani,  $ebcba\dots$  dizisinde ikinci  $b$ 'yi atlayarak  $ebca\dots$  dizisini oluşturmalıyız. Eğer  $cb$  ve  $ba$  kenarlarını silip, bu kenarlar yerine  $ca$  kenarını eklersek,  $b$ 'nin ikinci kez ziyaret edilmesini engellemiş oluruz ve gezgin  $(3+1)$  birimlik uzunluk yerine 3 birimlik uzunlukla  $c$ 'den  $a$  noktasına ulaşmış olur. Aynı yöntemle bütün dönüştürme işlemi tamamladığımızda, 22 birim uzunluğundaki  $ebcade$  Hamilton turunu elde ederiz. Bu yöntemle,  $G$  üzerindeki GSP'nin eniyi çözümünü bulmamıza rağmen, böyle bir sonuç maalesef her

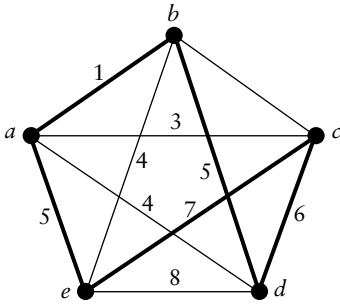


zaman garanti değildir. Ancak, bu sezgisel yöntem çok da kötü bir sonuç vermemektedir:

**Soru 5.** Üçgen eşitsizliğini sağlayan bir çizgede en kısa tur algoritmasıyla bulunan GSP turunun uzunluğunun, en kısa kapsayıcı ağacı oluşturan kenarların toplam uzunluğunun iki katından daha büyük olmayacağını gösterin.

Yukarıda anlattığımız iki sezgisel algoritma dışında, GSP literatüründe birçok sezgisel algoritma önerilmiştir. Bunların bir kısmı, incelediğimiz sezgisel algoritmalar gibi bir gezgin satıcı turu bulduktan sonra durur. Diğerleri ise verilen bir tur üzerinde değişiklikler yaparak, daha kısa turlar bulmaya yönelik “iyileştirici” girişimleri dener. Şimdi, bu türün en bilinen ve en yaygın kullanılan yöntemlerinden birini, turu oluşturan kenarların ikisini iki yeni kenarla değiştiren iyileştirme yöntemini inceleyelim.

**İki Kenar Değiştirme Algoritması.** İlk sorudaki çizge için verilen 24 uzunluğundaki *abdcea* Hamilton turu üzerinden bu algoritmayı tartışalım. Algoritmanın temel amacı, verilen bir tur üzerindeki komşu olmayan iki kenarı, iki yeni kenarla değiştirerek daha kısa bir tur bulmaktır. İki kenarı silinmiş bir turdan yeni bir tur elde etmenin bir tek yolu vardır. Şimdi, 24 birimlik *abdcea* turundaki 7 birimlik *ce* kenarını ele alalım. Diyelim ki bu uzun kenardan kurtulmak istiyoruz. Verilen turdan *ce*'siz yeni bir tur bulmak için, *ce*'yle birlikte *ce*'ye komşu olmayan 5 birimlik *bd* kenarından da ya da 1 birimlik *ab* kenarından da kurtulmamız gerekir. Bu durumda iki seçenekle karşı karşıyayız. Birinci seçenekte, (7+5) birimlik *ce* ve *bd* kenarlarını turdan silip (3+8) birimlik *bc* ve *ed* kenarlarını ekleyerek, 23 birimlik *abcdea* turunu elde ederiz. İkinci seçenekte, (7+1) birimlik *ce* ve *ab* kenarlarını (3+4) birimlik *ac* ve *be* kenarlarıyla değiştirerek, yine 23 birimlik ama başka bir tur olan *acdbea* turunu buluruz. Seçeneklerin ikisi de *abdcea*



değiştirerek daha kısa bir tur bulmaktır. İki kenarı silinmiş bir turdan yeni bir tur elde etmenin bir tek yolu vardır. Şimdi, 24 birimlik *abdcea* turundaki 7 birimlik *ce* kenarını ele alalım. Diyelim ki bu uzun kenardan kurtulmak istiyoruz. Verilen turdan *ce*'siz yeni bir tur bulmak için, *ce*'yle birlikte *ce*'ye komşu olmayan 5 birimlik *bd* kenarından da ya da 1 birimlik *ab* kenarından da kurtulmamız gerekir. Bu durumda iki seçenekle karşı karşıyayız. Birinci seçenekte, (7+5) birimlik *ce* ve *bd* kenarlarını turdan silip (3+8) birimlik *bc* ve *ed* kenarlarını ekleyerek, 23 birimlik *abcdea* turunu elde ederiz. İkinci seçenekte, (7+1) birimlik *ce* ve *ab* kenarlarını (3+4) birimlik *ac* ve *be* kenarlarıyla değiştirerek, yine 23 birimlik ama başka bir tur olan *acdbea* turunu buluruz. Seçeneklerin ikisi de *abdcea*

vardır. Şimdi, 24 birimlik *abdcea* turundaki 7 birimlik *ce* kenarını ele alalım. Diyelim ki bu uzun kenardan kurtulmak istiyoruz. Verilen turdan *ce*'siz yeni bir tur bulmak için, *ce*'yle birlikte *ce*'ye komşu olmayan 5 birimlik *bd* kenarından da ya da 1 birimlik *ab* kenarından da kurtulmamız gerekir. Bu durumda iki seçenekle karşı karşıyayız. Birinci seçenekte, (7+5) birimlik *ce* ve *bd* kenarlarını turdan silip (3+8) birimlik *bc* ve *ed* kenarlarını ekleyerek, 23 birimlik *abcdea* turunu elde ederiz. İkinci seçenekte, (7+1) birimlik *ce* ve *ab* kenarlarını (3+4) birimlik *ac* ve *be* kenarlarıyla değiştirerek, yine 23 birimlik ama başka bir tur olan *acdbea* turunu buluruz. Seçeneklerin ikisi de *abdcea*

turunu sadece 1 birim kısalttığı için, iki yeni turdan herhangi biri rastgele seçilebilir. Algoritma, yeni tur üzerindeki başka bir kenar çiftinin değiştirilmesiyle turun kısalıp kısalmadığını kontrol ederek devam eder. Eğer kısalma bulunursa, gerçekleştirilir. Daha kısa bir tur bulunamazsa, algoritma durur.

**Sonuç Yerine.** 1954'te 49 şehirlik GSP'nin çözümünden bu yana geçen zaman içinde, çözülen problemlerin büyüklüğü giderek artmış, günümüzde 15.000'den fazla şehir içeren problemlerin çözülebildiği bir birikime erişilmiştir. Çözüm zorluğu yüzünden, bir meydan okuma olarak görülen GSP'yle ilgili daha fazlasını merak eden okurlarımıza, merak turlarının başlangıç noktası için [LLKS]'yi ve [http://www.ing.unlp.edu.ar/ce-tad/mos/TSPBIB\\_home.html](http://www.ing.unlp.edu.ar/ce-tad/mos/TSPBIB_home.html) ve <http://www.dimacs.rutgers.edu/Challenges> sitelerini öneririz. ♦

2001'de Almanya'nın tüm 15.112 şehrini geçen ve her şehirden sadece bir kez geçen en kısa yol bulunmuştur. Bunun için Rice ve Princeton üniversitelerinin bilgisayarları 22 yıldan fazla çalışmışlardır... Toplam yol aşağı yukarı 66.000 km'dir, yani ekvatorun bir buçuk misli... İşte Almanya'nın en kısa yol haritası. İyi yolculuklar.

