**TABU SEARCH ALGORİTMASI**

Tabu Arama Algoritması, optimizasyon problemlerinin çözümü için F.Glover tarafından geliştirilmiş iteratif bir araştırma algoritmasıdır.

Temel yaklaşım, son çözüme götüren adımın dairesel hareketler yapmasını önlemek için bir sonraki döngüde tekrarın yasaklanması veya cezalandırılmasıdır. Böylece yeni çözümlerin incelenmesiyle Tabu Arama algoritması, bölgesel en iyi çözümün daha ilerisinde bulunan çözümlerin araştırılabilmesi için bölgesel-sezgisel araştırmaya kılavuzluk etmektedir.

Tabu Arama algoritmasının bölgesel optimalliği aşmak amacıyla kullandığı temel prensip, değerlendirme fonksiyonu tarafından her iterasyonda en yüksek değerlendirme değerine sahip hareketin bir sonraki çözümü oluşturmak amacıyla seçilmesine dayanmaktadır.

Bunu sağlamak amacıyla bir tabu listesi oluşturulur, tabu listesinin orijinal amacı önceden yapılmış bir hareketin tekrarından çok tersine dönmesini önlemektir.

Tabu listesi kronolojik bir yapıya sahiptir ve esnek bir hafıza yapısı kullanır. Tabu arama algoritması her ne kadar istenmeyen noktaların işaretlenmesi olarak açıklanmış olsa da daha cazip noktaların işaretlenmesi olarak ta kullanılır.

Yukarıdaki ifadeyi açıklarsak; amaç fonksiyonu c(x) maliyet veya kar fonksiyonun en küçük veya en büyük değerin aranmaktadır fakat bu aramada x vektörü ile belirtilen kısıtlamalara uyularak çözüme ulaşılacaktır.

Başka bir ifade ile her x elemanı bir hareketi temsil eder ve tüm hareketler X ile gösterilmektedir. Ancak daha doğru bir varsayım x vektörlerinin TA bellek yapısı olarak kullanıldığıdır. Böylece vektörde tutulan bellek değerine bağlı olarak çözüm aramada bazı hareketler tabu olarak kabul edilip engellenecek, bazılarına ise daha fazla odaklanacaktır.

X vektöründeki her bir hareket ise mevcut çözümün bir komşusunun seçimini temsil eder.

**TABU ALGORİTMASININ ÇALIŞMA PRENSİBİ**

**Başlangıç çözümünün oluşturulması:**

En genel şekilde başlangıç çözümü rastsal olarak elde edilir. Ancak ilgilenilen, problem için geliştirilmiş olan bir sezgisel algoritmadan yararlanarak da başlangıç çözümünden elde edilmesi mümkündür.

**Hareket Mekanizması :**

Mevcut bir çözümde yapılan bir değişiklikle yeni bir çözümün elde edilmesi hareket mekanizmasıyla gerçekleştirilir. Hareket mekanizmasındaki olası hareketler, mevcut çözümün komşularını oluşturur.

**Komşuluk:**

Tabu Aramada en önemli bileşenlerden birisi de komşuluk yapısıdır. Çözümü iyileştirmek için amaç fonksiyonun değeri açısından en iyi hareketlerin seçilmesidir.Komşulukların oluşturulmasında seçilen komşuluk üretme yapısına göre problem boyutu n olduğu durumda (n-1) tane komşuluk üretilir.

**Hafıza:**

Tabu Search algoritmasının temel elemanlarından biride hafızadır.Arama boyunca ortaya çıkan durumlar, H hafızasına kayıt edilir. Yapılmasına izin verilmeyen hareketler "tabu" olarak adlandırılır ve esnek hafıza içinde "tabu listesi" adı altında kaydedilirler. Bu hareketler belli bir süre sonra tabu listesinden çıkarılır ve yapılmasına izin verilir.

**Hafıza Türleri:**

Yakınlık Geçmişe dayalı Bellek Yapısı (Kısa Dönem Hafıza)(YBY)

En temel TA bellek yapısıdır. Esas olarak görevi yakın geçmişte yapılan hareketlerin bir süre için tabu olarak işaretlemektir. Burada kast edilen süre tabu süresi yani çevrim sayısıdır. Belirlenen t değeri süre kadar hareket tabu olarak belirlenir ve daha sonra belirlenen hareket tabu olmaktan çıkarılır. Örneğin t= 5 olacak şekilde statik olarak değer verilebildiği gibi t değeri mint ve max t sınırları arasında sistematik veya rastgele seçilebilir.

**Tabu Listesi :**

Tabu listesi, araştırmanın herhangi bir iterasyonu içinde hangi seçimlerin tabu grubu olmak zorunda olduğu ve kaç tanesi üzerinde karar verme ve tabu listesini güncelleme ile ilgilenir. Tabu listesinin boyutu, sonucu önemli derecede etkileyebilir. Deneysel sonuçlar problemin boyutu büyüdükçe problem boyutu ile orantılı olarak tabu listesi uzunluğunun da büyümesi gerektiğini göstermiştir.

**Tabu Yıkma Kriterleri:**

Tabu yıkma kriterleri,tabunun ortadan kalkabileceği durumları ifade etmektedir. En genel tabu yıkma kriteri, mevcut durumdan daha iyi bir sonuç verecek tabu hareketinin yapılmasına izin verilmesidir. Bu kriterin kullanılması TA algoritmasının etkinliğini artırmaktadır. Ayrıca, eğer tüm mümkün hareketler tabu ise bu hareketlerden tabu süresinin bitmesine en yakın olan bir tabu hareketine izin verilir.

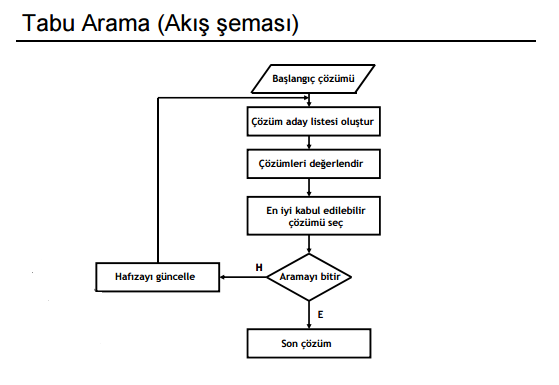
Durdurma Koşulu bir veya birden fazla durdurma koşulunu sağlayıncaya kadar aramasını sürdürmektedir. Bu koşullardan bazıları aşağıda verilmiştir.

- Seçilen bir komşu çözümün komşusunun olmaması

- Belirli bir iterasyon sayısına ulaşılması

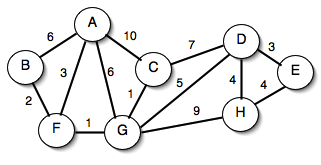
- Belirli bir çözüm değerine ulaşılması

- Algoritmanın bir yerde tıkanması ve daha iyi sonuç üretememesi

****

TA algoritması, bir başlangıç çözümü ile aramaya başlar. Algoritmanın her iterasyonunda tabu olmayan bir hareket ile mevcut çözümün komşuları içerisinden bir tanesi seçilerek değerlendirilir. Eğer amaç fonksiyonunun değerinde bir iyileştirme sağlanmışsa komşu çözüm, mevcut çözüm olarak dikkate alınır. Seçilen bir hareket tabu olmasına rağmen tabu yıkma kriterlerini sağlıyorsa, mevcut çözümü oluşturmak için uygulanabilir. Geriye dönüşleri önlemek için, bir takım hareketler tabu listesine kaydedilerek tekrar yapılması belirli bir süre için yasaklanır. Belirlenen bir durdurma koşuluna göre algoritmanın çalışması sonlanmaktadır.

**Tabu Search Algoritmasının Minimum Spanning Tree Problemine Uygulanması**



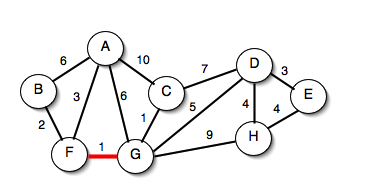
Verilen örnek graph üzerinde tabu search algoritması kullanılarak MST probleminin çözülmesini inceleyelim:

**Başlangıç Noktasının Belirlenmesi:**

Öncelikle bir baslangıç noktası rastsal olarak belirlenir. Biz örneğimizde F noktasını başlangıç olarak alalım. Başlangıç noktasının başka bir nokta olarak alınması MTS çözümünü değiştiremeyecektir.

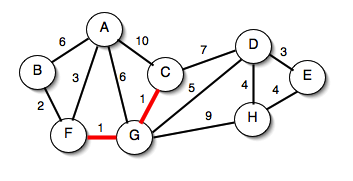
**Hareket Mekanizması:**

**1) F noktasından gidilebilecek düğümler seçilir.Bunlar FB,FG,FA dır.Bunların hepsi hafızaya atlıır. Tabu Listemize FB ve FA eklenir**.



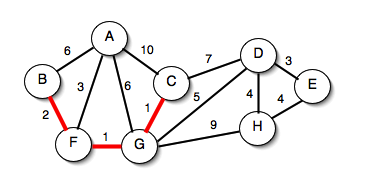
En düşük maliyetli edge FG dir.

**2)Şimdi de G’den gidilebilecek düğümler belirlenip hafızaya atılır.Bunlar GA,GC,GD,GH tır.**



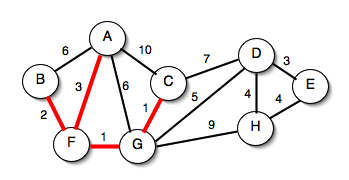
En düşük maliyetli yol GC olarak secilir.Diğerleri tabu listesine eklenir.

**3)C noktasından gidilebilecek düğümler belirlenir.Bunlar CA, CD ‘dir.Bunlarda digerlerinin yanına hafızaya eklenir.**



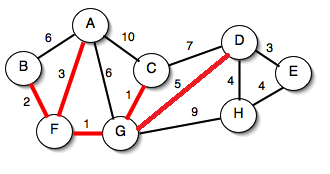
En düşük maliyetli yol hafızadan seçilir. Eğer seçilen yol tabu listesinde de yer alıyorsa yani aşamada olduğu gibi FB esnek hafızadan(tabu listesinden ) silinir.CA ve CD tabu listesine eklenir.

**4)Şu an B noktasından gidilebilecek yerleri belirliyoruz.Bu BA’dır.Hafızaya eklenir.**



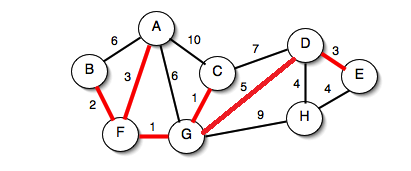
Hafıza komple taranır ve en küçük maliyetli yol FA olarak bulunur.BA tabu listesine eklenir.FA tabu listesinden silinir.

**4)A noktasından gidilebilecek yollar belirlenir.Daha önceden belirlenen yollar haricinde AC hafızaya eklenir.**



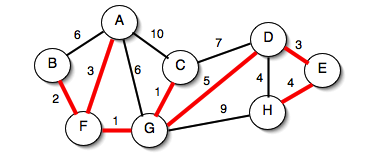
Hafıza tekar taranır .GD en küçük maliyetli yol olarak belirlenir.GD tabu listesinde olduğu için GD tabu listesinden silinr.

**5)Dnoktasından gidilebilecek yerler DE ve DH olarak belirlenir.Bunlar hafızaya atılır.**



En küçük maliyetli yol hafızadan DE olarak bulunur. DH tabu listesine atılır.

**6) E noktasından gidilebilecek yerler EH belirlenip hafızaya yazılır.**



Hafızadan en düşük maliyetli yol EH ve DH dan biri seçilir. Biz EH ı en düşük maliyetli yol olarak seçelim.

Yukarıdaki örnekte tabu yıkma kriteri tabu listesinde yer alan en düşük maliyetli yolun esnek hafızadan(tabu listesinden) silinmesidir.

En son gelinen nokta H’den gidilebilecek yollar HD ve HG dir.Bu yollar tabu listesinde ve hafızada bulunduğundan bitiş noktası H olarak bulunmuş olur.

Tabu listesindeki yollar bize gidilmemesi gerek yolları verir.Onun dışında graphda yer alan bütün yollar bize Minimum maliyetli yolu verir.

**Mininmum Path F🡪G**

**G🡪C**

**F🡪B**

**F🡪A**

**G🡪D**

**D🡪E**

**E🡪H**

**PROBLEMİN ÇÖZÜM AŞAMALARI**

**1-Yol bilgisini tutacak Array in ve bileşenlerinin tanımlanması:**

8\*8 LİK matrisimizde herhangi bir Node’dan başka bir Node’a yol varsa yol maliyeti degeri o satır ve sütün bilgisinde tutulmaktadır.Raporumuzda yer alan

A,B,C,D,E,F,G,H edgeleri

A 🡪 1.satir,1.sütun

B 🡪2.Satir ,2.sütun

C🡪3.satir, 3.sütun

.

.

.

H 🡪8.satir,8. sutün bilgisi olarak tutulacaktır.

Örneğin 6. satır 7. sütünda F'den G ye gidilen yol var ve 1 birim maliyetlidir.Bir başka örnek A’dan B ye yol var ve 6 birim maliyetlidir.

Arrayde de görüldüğü üzere aynı Node’dan kendine yol bulnmamaktadır.

C🡪C , D🡪D yol bulunmadığından yol maliyeti 0’dır.

**array**= [ 0 6 9 0 0 3 6 0;

6 0 0 0 0 2 0 0;

9 0 0 7 0 0 1 0;

0 0 7 0 3 0 5 4;

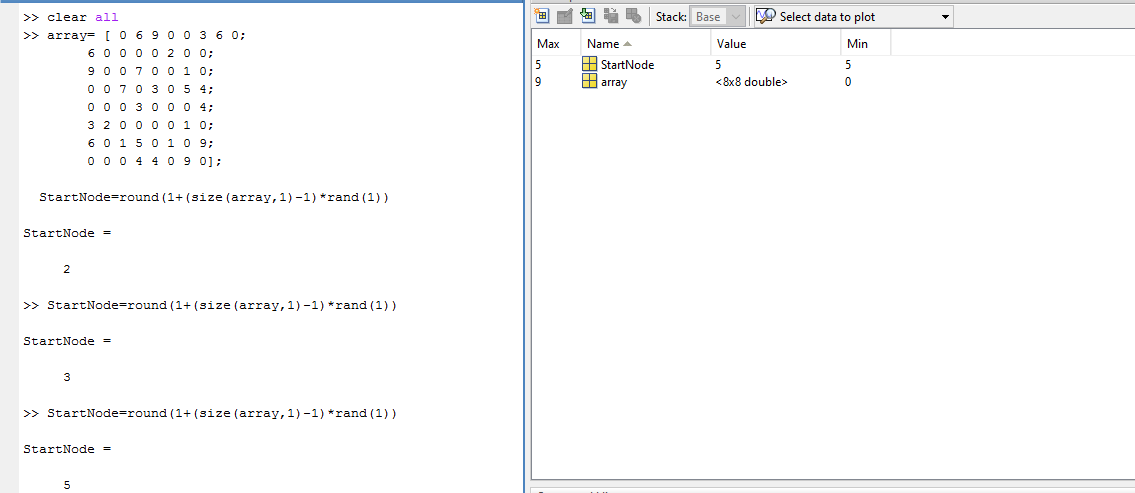
0 0 0 3 0 0 0 4;

3 2 0 0 0 0 1 0;

6 0 1 5 0 1 0 9;

0 0 0 4 4 0 9 0];

**2-Başlangıç Node’nun Belirlenmesi:**Satırlarımızdan herhangi biri başlangıç nodu olarak seçilecektir.Bunu random olarak yapacaktır.Rastsal olarak belirlenen ilk node yol maliyetini değiştirmeyecektir.

****

**3- Belirlenen ilk hareket noktası için o satırdaki 0 dan farklı elemanların indis numaraları(indexarray ‘de tutulacak) ve yol maliyet değeri lerinin(values) bulunması** **ve kullanılacak hafıza için boş dizilerin tanımlanması:**

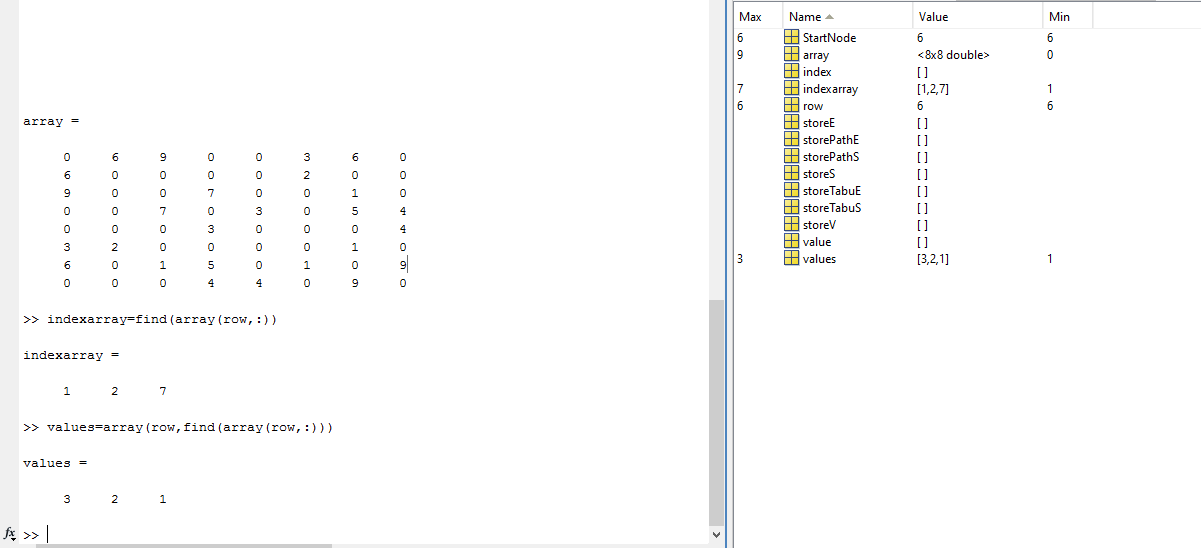
Resimde de görüldüğü üzere F Node’u için 1. 2 ve 7.ini indislerde eleman vardır.Ve bunların değerleri de sırasıyla 3 2 ve 1 dir resimde gördüğümüz komutlar bu işlemleri gerçekleştirmektedir.

Üç farklı boş dizi tanımlanacaktır.

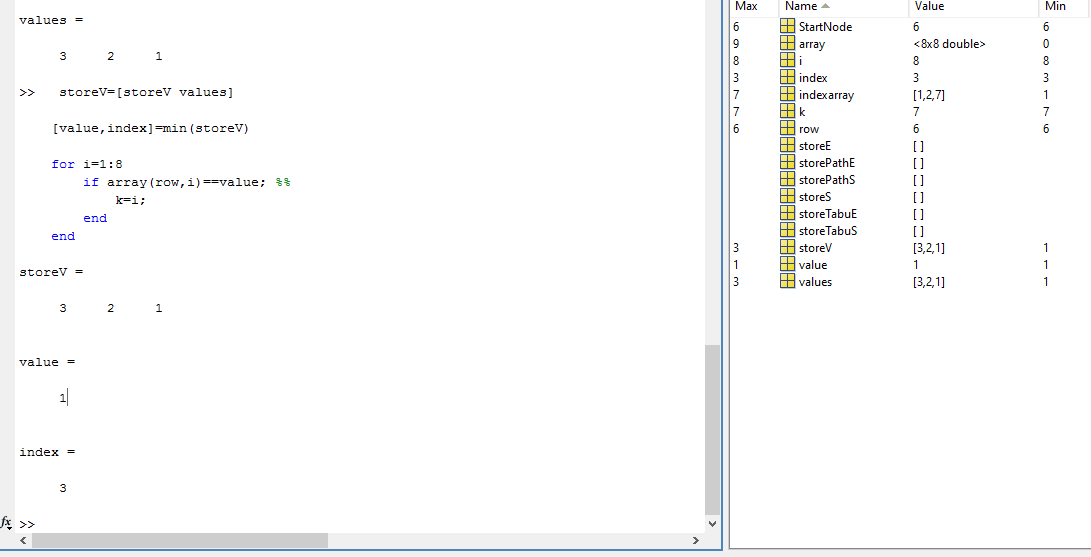
Birincisi tüm elemanların tutulduğu storeS,storeE hem x hem y kordinatlarını tutmak için iki farklı dizi tanımladık.

storeTabuE,storeTabuS gidilmeyen yolların x y koordinetlarının tutulacağı dizidir.

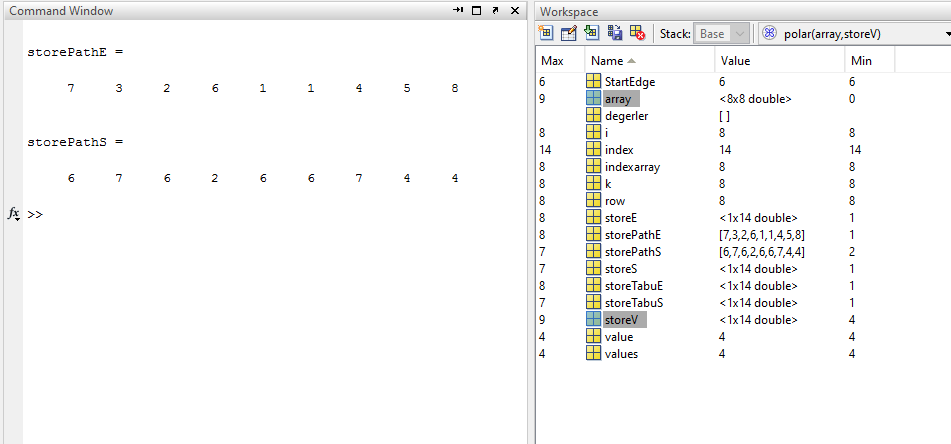
storePathS,storePathE minimum yolların x y koordinatlarının tutulacaı dizidir.

****

**4-) Tüm hafıza tutugumuz değerler arasında minimum olanın indis numarası ve değeri bulunur.**

****

**5)Daha sonra bulunan minimum yoldan gidilen sütün satır gibi düşünülerek aynı işlmler o adım için gerçekleştirilir. Yani başlangıç nodemuz 6 idi minimum yolu maliyetini 7. İndisteki 1 birim maliyet olarak bulduk. Şimdi de yedinci satırdan gidilecek yollar bulunup bunlar tekrardan hafızaya atılıp mininmum yol tekrardan bulunup StorePath e atılacaktır. StoreTabu ya da gidilmeyen yollar atılacaktır. Bulunan tüm yollar da eğer daha önce hafıza yoksa hafızaya kaydedilecektir.Bu işlem StorePath değeri 8 olana kadar yani tüm nodeler dolaşana kadar gerçekleşecektir (while(size(StorePath<9) kontrolu bunu sağlamaktadır.Her iterasyonda StoreTabu ve Store güncelllenecektir.**

****

**Görüldüğü üzere gidilen yol üzergahı (F🡪G) (G🡪C) .........(D🡪H) şeklinde gerçekleşmektedir.**

**KAYNAKÇA:**

1)[**http://www.geneseo.edu/~baldwin/csci242/spring2013/0404mst.html**](http://www.geneseo.edu/~baldwin/csci242/spring2013/0404mst.html), Minimum Spanning Trees, Thursday April 4, CSci 242, Spring 2013  
Prof. Doug Baldwin

2)[**http://acikerisim.pau.edu.tr/xmlui/bitstream/handle/11499/1256/%C5%9Eahin%20Bayzan.pdf?sequence=1&isAllowed=y**](http://acikerisim.pau.edu.tr/xmlui/bitstream/handle/11499/1256/%C5%9Eahin%20Bayzan.pdf?sequence=1&isAllowed=y)**,** T.C. PAMUKKALE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ, ARAÇ ROTALARININ EN KISA YOL ALGORİTMALARI KULLANILARAK BELİRLENMESİ VE .NET ORTAMINDA SİMÜLASYONU, Şahin BAYZAN

3)[**http://bilgisayarkavramlari.sadievrenseker.com/2007/12/24/asgari-tarama-agaci-en-kisa-orten-agac-minimum-spanning-tree/**](http://bilgisayarkavramlari.sadievrenseker.com/2007/12/24/asgari-tarama-agaci-en-kisa-orten-agac-minimum-spanning-tree/)

Asgari Tarama Ağacı (en kısa örten ağaç, minimum spanning tree), Sadi Evren Şeker, Aralık 24, 2007

4**)**[**https://www.mathworks.com/help/bioinfo/ref/graphallshortestpaths.html**](https://www.mathworks.com/help/bioinfo/ref/graphallshortestpaths.html)

Minmum yolun bulunup bulunmadığını kontrol etmek için graphallshortestpath methodundan test edillmesi için yarlanıldı.

5) <https://www.ics.uci.edu/~eppstein/161/960206.html>

6) <http://freesourcecode.net/matlabprojects/58814/tabu-search-matlab-code#.WRm_F2jyjIU>