Journal of Turkish

Operations Management

İki amaçlı en kısa yol problemi ve bir uygulaması

Hakan Gürsoy*, Ekrem DUMAN[2]
[1]Hacettepe Üniversitesi İşletme Bölümü, Beytepe Kampüsü Çankaya Ankara
e-mail,hg2324@columbia.edu ORCID No: http://orcid.org/0000-0002-7759-1502
[2]Özyeğin Üniversitesi Mühendislik Fakültesi Nişantepe Mah. Orman Sok. İstanbul
e-mail, ekrem.duman@ozyegin.edu.tr ORCID No: http://orcid.org/0000-0001-5176-6186
*Sorumlu Yazar

Makale Bilgisi

Makale Geçmişi:
Geliş: 18.02.2022
Revize: 04.06.2022
Kabul: 25.11.2022

Anahtar Kelimeler:

En kısa yol problemi
İki amaçlı en kısa yol problemi
Martins algoritması
Etiketleme algoritmaları
A* algoritmaları

Özet

Bu çalışmada çok amaçlı en kısa yol probleminin özel bir durumu olan iki amaçlı en kısa yol problemi incelenmektedir. Gerçek hayatta sıkça karşılaşılan bu problem için literatürde birçok çözüm metodu önerilmiştir. Problemin çözümü tek bir sonuç değil bir çözüm kümesinden oluştuğu için bu metotlar uzun hesaplama ve işlem sürelerine sahiptir. Bu nedenle de literatürde çözüm kümesinin tamamını bulma garantisi olmasa da çok daha kısa sürede çözüm üreten sezgisel metotlar geliştirilmiştir. Bu çalışmada çözüm kümesinin tamamını veren metotlar ile sözü geçen sezgisel metotlar açıklanacak ve bu metotlar arasında karşılaştırma imkânı sunan bir uygulama yapılacaktır.

Bicriteria Shortest Path Problem and an Application

Article Info

Article History:
Received: 18.02.2022
Revised: 04.06.2022
Accepted: 25.11.2022

Keywords:

Shortest Path Problem
Bicriteria Shortest Path Problem
Martins Algorithm
Labeling Algorithms
A* Algorithms

Abstract

In this paper bicriteria shortest path problem, a specific case of shortest path problem, is studied. This problem can be seen frequently in real life, and there are methods that can produces solutions to the problem. Since the solution of the problem is a set of solution points, these methods need long computation times. Thus heuristic solutions, were also suggested,which does not guarantee to find the complete set of solutions, but soslving the problem in a relatively shorter time compared to exact methods,. In this paper we will explain both of exact methods and heuristic methods and describe an application to that compares them.

DOI: 10.56554/jtom.1075286

1. Giriş

En kısa yol problemi insanlığın ortaya çıkışından beri sürekli çözmeye çalıştığı problemlerden birisidir. Bu problemi kısaca, bir noktadan bir noktaya, belirli şartları sağlayarak en kısa mesafede varma problemi olarak tanımlamak mümkündür. Doğası gereği insan mümkün olduğunca en kısa yolu seçer.

En kısa yol problemi sadece güzergah hesaplama ya da rotalama problemlerinde değil, bir çok gerçek hayat probleminde de kullanılmaktadır. Bunlara örnek olarak robot navigasyonu, trafik planlama, mikroişlemci elektrik akım problemleri, ağlarda ip ayarlaması ve finansal piyasalarda arbitraj imkanlarının bulunması verilebilir (Ahuja v.d.,1988; Sedgewick, 2014).

Ek kısa yol probleminin çözümü için yapılan çalışmaların geçmişi 1950 ve 1960'lara dayanmaktadır. Bu çalışmalar sonucunda literatürde öncü olmuş 3 farklı metot geliştirilmiştir. Bunlar Bellman-Ford metodu (Bellman,1958; Ford,1956), Dijkstra metodu (Dijkstra, 1959) ve Floyd-Warshall metodudur (Floyd, 1962). Dijkstra'nın metodu bir düğümden başka bir düğüme en kısa yolun bulunması için geliştirilmiş hızlı bir algoritmadır. Uygun bir veri yapısı (heap gibi) kullanılması durumunda O(N log N) süre karmaşıklığına sahiptir, ancak algoritmanın çalışabilmesi için kenarların negatif olmaması gereklidir. (Sedgewick, 2014). En kısa yol problemlerinin ilk çözümlerinden olan Bellman-Ford metodunda ise negatif kenarların olması durumunda da çözüm bulunabilmektedir. (Sedgewick, 2014). Floyd-Warshall metodunda ise bütün düğümlerden bütün düğümlere en kısa yol bulunabilmektedir.

İki amaçlı en kısa yol problemi ise en kısa yol probleminin özelleştirilmiş bir halidir. İki amaçlı en kısa yol probleminde, genellikle birbiri ile çelişen (maliyet ve zaman gibi) iki farklı amaçta en kısa güzergahlar aranmaktadır. Tehlikeli madde taşımacılığı gibi bazı problemlerde ise amaçlardan bir tanesi en kısa yol olurken diğeri, en tehlikesiz yol olabilmektedir. Bu durumda ise ikinci amaçta toplama değil, en tehlikesiz yolun bulunması hedef olmaktadır.

Problemi detaylı olarak incelemeden önce literatürde problem hakkında kullanılan iki kavramı tanımlamak gerekmektedir. Bunlardan ilki çözümün etkin olmasıdır. Etkin bir çözüm, kendisinden bütün amaçlar açısından daha iyi bir çözümün bulunmadığı bir çözümdür. Yani başka bir çözüm kümesi bazı amaçlar açısından daha iyi olabilir, ancak en azından bir amaç açısından daha iyi veya eşit sonuç vermeyecektir. (Skriver, 2000a). Bu çözümlere aynı zamanda Pareto optimal çözümler de denilmektedir. İkinci önemli kavram ise çözümün diğerleri tarafından domine edilip edilmediğidir. Domine edilme kavramı literatürde özellikle tanımlanmış ve önem verilmiş bir kavramdır. Bir çözüm etkin çözüm ise bu durumda domine edilmemiş bir çözümdür. Domine edilmiş olan çözümler etkin çözüm değildir. Domine edilmemiş çözümler ise mutlaka etkin çözüm olmak zorundadır (Skriver, 2000a).

Çok amaçlı en kısa yol problemlerinin çözümü için araştırmalar 1970'lerde başlamıştır. 2000'li yılların başına gelindiğinde literatürde çeşitli yöntemler için bir bilgi birikimi oluşmuştur. Bu minvalde Skriver (2000a) literatürdeki kesin metotları çalışmasında açıklamıştır. Kesin metotlar etiketleme ve ağaç/güzergah yöntemleri olarak ikiye ayrılmaktadır. Bunlardan etiketleme yöntemleri Hansen'in (1980) çalışması ile başlamıştır. Ancak esas bu konudaki ana çalışma Martins'in (1984) çalışması ile ortaya koyduğu Martins algoritmasıdır. Etiketleme yöntemleri literatürde pek çok çalışmada yer almıştır. Daha az kullanılan ağaç/güzergah yöntemleri ise iki aşamalı arama ve k-en kısa yol algoritmalarıdır.

Serafini'ye (1986) göre iki amaçlı en kısa yol problemi NP-Complete (polinomsal sürede çözümü olmayan) bir problemdir. Ancak eldeki çizgelerde döngüler bulunmuyorsa, polinomsal sürede çözülebilecektir (Ehrgott, 2005). Öte yandan polinomsal sürede çözülebilmesi de çok hızlı çözüleceği anlamına gelmemektedir. Zaten literatürde yapılan uygulamalarda çözüm sonuçlarının oldukça uzun zaman aldığı da görülmüştür. Bu nedenlerle yukarıda açıklanan kesin metotlar haricinde, literatürde çok amaçlı en kısa yol probleminin çözümü için çeşitli sezgiseller de önerilmiştir. Aşağıda literatürde yer alan sezgisellerden ulaştıklarımız yer almaktadır.

En kısa yol algoritmalarında kullanılan sezgisellerin en bilineni A* algoritmasıdır. Bu algoritma ile her düğüm noktasına bir sezgisel fonksiyon değeri atanmaktadır. Arama ağacı oluşturulurken, bu sezgisel değeri ile o ana kadar alınan mesafenin toplamına göre bir A* fonksiyon değeri oluşturulmaktadır. En küçük A* değerleri takip edilerek, en kısa yol bulunabilmektedir. Literatürde yapılan çalışmalarda, yukarıda sözü geçen sezgisel fonksiyonunun belirli kriterleri taşıması durumunda, A* algoritmasının kesin sonuca ulaşacağı vurgulanmıştır. Çok amaçlı en kısa yol probleminin çözümünü hedefleyen sezgisellerin ilk grubu A* algoritmasını temel alarak oluşturulmuştur. Burada öne çıkan algoritmalar MOA*, NAMOA*, NAMOA*dr ve BOA* algoritmalarıdır. MOA* algoritması yukarıda sözü geçen A* tabanlı algoritmaların ilkidir (Stewart ve White,1991). NAMOA* ise sezgisel fonksiyonunda değişiklik yapmak suretiyle MOA* algoritmasını geliştirmiştir (Mandow ve Cruz., 2010). NAMOA*dr algoritması ise, NAMOA* algoritmasını oluşturan araştırmacılar tarafından, geçilmeyecek olan güzergahların elenmesi yoluyla, problemin boyutunun küçültülmesine yönelik bir geliştirme algoritmasıdır (Mandow, Pulido, Cruz, 2015). Bu algoritmaların tamamı çok amaçlı en kısa yol problemlerinin tamamında uygulanabilecek algoritmalardır. BOA* algoritması ise sadece iki amaçlı en kısa yol problemi için geliştirilmiştir (Ulloa, Yeohz, Baier, Zhang, Suaz, Koenig, 2020). Bu algoritma, daha önceden oluşturulan algoritmalara göre oldukça hızlı sonuç vermektedir.

DOI: 10.56554/jtom.1075286

A* tabanlı sezgiseller dışında da genel arama algoritmalarında kullanılan meta-sezgisellerin iki amaçlı en kısa yol problemine uygulanmasını içeren çalışmalar da mevcuttur. Bunlara örnek olarak damlacık (ripple) algoritması verilebilir (Hu, Gong, Ming, Zhang, Li, Leeson, Liao, 2020). Damlacık algoritması, su damlalarının suyun içerisinde dağılmalarını taklit eden bir algoritmadır. Tek amaçlı arama problemlerinde sıkça kullanılan bir algoritmadır. İki amaçlı en kısa yol probleminde de uygulaması mevcuttur. Damlacık algoritması haricinde genetik algoritma tabanlı çalışmalar da mevcuttur (Cheikh, Bassem, Taicir 2010).
Yukarıda belirttiğimiz yöntemleri literatürdeki sınıflandırmaya göre özetlemek için Tablo 1 kullanılmıştır.

Tablo 1. Çözüm Yöntemleri Sınıflandırma Tablosu

Kesin Yöntemler		**Sezgiseller**	
Etiketleme Yöntemleri	Güzergah/Ağaç yöntemleri	A* tabanlı yöntemler	Diğer sezgiseller
Etiket Belirleme	*İki aşamalı çözüm*	*MOA** *NAMOA** *BOA**	*Genetik Algoritma* *Damlacık Algoritması*
Etiket Düzeltme	*K-En Kısa Yol*		

Çalışmamız, literatürde çok amaçlı en kısa yol problemine ilişkin olarak kesin yöntemler ve sezgiselleri bir arada açıklayan ve bu kapsamda yapılmış çalışmaları en geniş şekilde kapsamaya çalışan bir makaledir. Bildiğimiz kadarıyla hem kesin yöntemler hem de en yeni sezgisel yöntemleri bir arada kapsayan bir yayın bulunmamaktadır. Çalışmamızın ikinci katkısı ise bu yöntemlerin literatürde sıkça kullanılmış olan veri setleri kullanılarak karşılaştırılmasıdır. Bu karşılaştırma sonucunda hem geleceğe dönük araştırma konuları ortaya çıkmakta hem de yöntemleri kullanacak olan uygulamacılara öneriler getirilmektedir.
Konu hakkında genel bir tanımlama ve literatür taraması yapılan bu bölümden sonra, çalışmanın ikinci kısımda problem daha detaylı anlatılacak ve model yer alacaktır. Bundan sonra gelen üçüncü bölümde çözüm yöntemlerinin sınıflandırılması ve kısaca tanımlanmasına ayrılmıştır. Bu yöntemlere ilişkin uygulamanın ve karşılaştırmanın yapıldığı dördüncü bölümün ardından sonuç bölümü ile makale tamamlanacaktır.

2. Problemin tanımlanması

Çalışmamızın giriş bölümünde kısaca değinildiği üzere iki amaçlı en kısa yol problemi, tek amaçlı en kısa yol problemini temel almaktadır. Ancak burada çözüm kümesi tek bir nokta değil, bir pareto yüzeyi olmaktadır.
Aşağıda Tablo-2'de iki amaçlı en kısa yol problemi için matematiksel gösterim yer almaktadır. G=(N,E) şeklinde bir çizge (graph) olsun. N={1,..,n} düğümlerin (node) oluşturduğu kümeyi, E={(i,j) | i,j ∈ N} ise kenarların (edge) gösteren kümedir. E kümesinin elemanlarının uzunlukları ise w_{ij} (i,j ∈ N) olarak gösterilecektir. Başka bir söyleyişle, w_{ij} , i düğümünden j düğümüne olan bağlantının uzunluğudur. Aşağıdaki gösterimde ikinci amaç için ağırlıklar ise v_{ij} olarak gösterilmektedir. Burada v_{ij} değişkenlerinde yer alan ağırlıklar iki amaçlı en kısa yol problemine ilişkin amaçların çeşidine göre farklı nitelik alabilir. Örneğin ikinci amaç en kısa süre ise v_{ij} değeri bir süre değeri, ikinci amaç maliyet ise v_{ij} parasal bir değer olacaktır. *bas* ve *bit* ise sırasıyla başlangıç ve bitiş düğümlerini göstermektedir. Gösterimdeki "Min", pareto optimalliği temsil etmektedir (Shi, Zhou, Wang, Tao , Liu.2017).

Tablo 2. Matematiksel Model

Amaç Fonksiyonu:

Min

$$\sum_{i,j \in E} x_{ij} w_{ij} \tag{1}$$

$$\sum_{i,j \in E} x_{ij} v_{ij} \tag{2}$$

Kısıtlar:

$$\sum_{j \in N} x_{ij} - \sum_{j \in N} x_{ji} = 0 \;\; \forall\, i \;\in N - \{bas, bit\} \tag{3}$$

$$\sum_{j \in N} x_{bas\ j} - \sum_{j \in N} x_{j\ bas} = 1 \tag{4}$$

$$\sum_{j \in N} x_{bit\ j} - \sum_{j \in N} x_{j\ bit} = -1 \tag{5}$$

DOI: 10.56554/jtom.1075286

$$w_{ij}, v_{ij} \geq 0 \ , \forall \ (i,j) \ \in E \quad (6)$$

$$x_{ij} \in 0{,}1 \ , \forall \ (i,j) \ \in E \quad (7)$$

Matematiksel modelde iki amaç fonksiyonu bulunmaktadır. İlk amaç fonksiyonu (1 numaralı fonksiyon) birinci amacın minimize edilmesini hedeflemektedir. İkinci amaç fonksiyonu (2 numaralı fonksiyon) ise ikinci amacın minimize edilmesini hedeflemektedir. İki amaç fonksiyonu da benzer şekildedir; her kenara ilişkin amaç ağırlık değeri ile o amacın çözümde yer alıp almayacağını belirleyen x_{ij} değişkeninin çarpımlarının, bütün kenarlar için toplanması ile oluşan fonksiyonlardır.

Kısıtlara gelinirse, ilk kısıt (3 numaralı denklem) bir akış koruma kısıtıdır. Başlangıç ve bitiş düğümleri hariç herhangi bir düğüme giren ve o düğümden çıkan akış eşit olmalıdır. İkinci kısıt (4 numaralı denklem) başlangıç düğümü için akış koruma kısıtını vermektedir. Bu kısıta göre, başlangıçtan çıkan akış ile başlangıca giren akış arasında bir fark olmalıdır. Bu durumun tersi ise üçüncü kısıtta (5 numaralı denklem), yani bitiş düğümü için akış koruma kısıtında yer almaktadır. Dördüncü kısıt (6 numaralı denklem), kenarların ağırlıklarının negatif olmamasını gerektiren kısıttır. Bu kısıttan anlaşılabileceği üzere problemimizde negatif ağırlıklar yer almamaktadır. Ancak bilindiği üzere, tek amaçlı en kısa yol probleminde negatif ağırlıkların yer aldığı modeller ve bu modelleri çözen metotlar bulunmaktadır. Örneğin tek amaçlı en kısa yol probleminde Dijsktra'nın (1959) algoritması negatif ağırlıklara izin vermezken Bellman-Ford (Bellman,1958; Ford,1956) ve Floyd-Warshall (Floyd, 1962) algoritmaları negatif ağırlıklara (negatif ağırlıklı döngüler olmaması şartıyla) izin vermekte ve çözüm üretmektedir (Sedgewick, Wayne, 2014). Çift amaçlı en kısa yol algoritmasının negatif ağırlıklara izin verecek şekilde genişletilmesi de mümkündür. Son kısıt (7 numaralı denklem)ise herhangi bir kenarın çözümde yer alıp almadığını belirleyen x_{ij} değişkeninin ikili (binary) değişken olduğunu belirleyen kısıttır.

Yukarıda da belirtildiği üzere, bu modelin çözümü bir pareto yüzey ortaya koymaktadır. Pareto yüzeyde, domine edilmemiş bütün çözümler yer alacaktır. Tablo-3'te örnek bir çözüm kümesi ile domine edilmiş ve edilmemiş çözümler gösterilecektir.

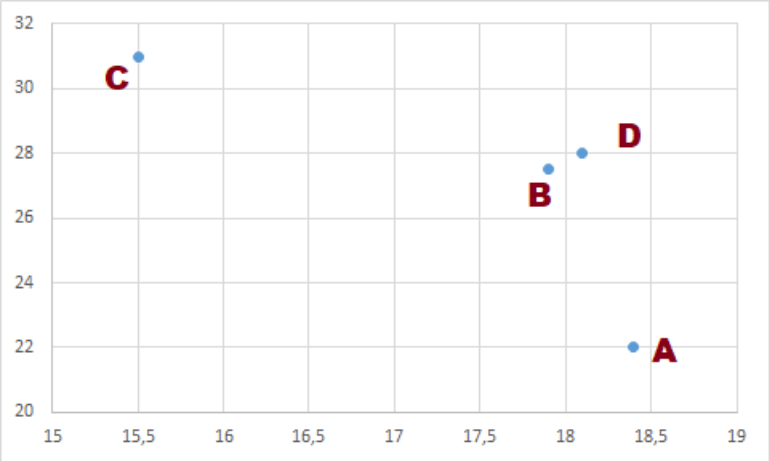
Tablo 3. Örnek Çözüm Kümesi

Güzergahın Adı	Mesafe (km)	Ortalama süre (dk)
A	18,4	22
B	17,9	27,5
C	15,5	31

Bu güzergahların herhangi birisinin diğerleri tarafından domine edilip edilmediği literatürde özellikle tanımlanmış ve önem verilmiş bir kavramdır. Bir çözüm, diğer çözümler ile karşılaştırıldığında bütün amaçlar açısından daha kötü sonuç veriyorsa etkin olmayan bir çözümdür, aksi halde etkin bir çözümdür. Bir çözüm etkin çözüm ise bu durumda domine edilmemiş bir çözümdür. Domine edilmemiş çözümler ise mutlaka etkin çözüm olmak zorundadır (Skriver , 2000a). Örneğimizde, A güzergahı en kısa süreye sahiptir, C güzergahı ise en kısa mesafeye, B güzergahı ise A'dan daha kısa mesafeye sahipken C'den daha kısa sürede bitmektedir. Bu üç güzergâh da diğerlerini domine edememektedir, zira her iki amaçta de diğer herhangi bir güzergahtan daha üstün olan bir güzergâh bulunmamaktadır. Bu örneğimizde farazi bir D güzergahı mevcut olsa (18,1 km uzunluğunda ve 28 dk'da tamamlandığını varsayalım) bu güzergâh her ne kadar A ve C güzergahları tarafından domine edilmese de B güzergahı tarafından domine edilmektedir çünkü B güzergahı, D güzergahına göre hem daha kısa mesafeye sahiptir hem de daha kısa sürede tamamlanmaktadır. Bu nedenle B güzergahının bulunduğu bir güzergâh listesinde D güzergahı yer alamayacaktır.

Yukarıdaki örneği bir de eksenlerinde amaçların yer aldığı bir serpme diyagramında (Şekil-1) gösterelim. Aşağıda yer alan diyagramda, D noktası B noktasına göre hem yukarıda hem de sağda olduğundan domine edilmiş durumdadır. Diğer noktalar ise, daha aşağı ve daha sollarında bir nokta olmadığından domine edilmiş olmayacaktır (domine edilmemiş noktalardır). Domine edilmemiş olan noktalara pareto optimal noktalar, bu noktaların tamamının oluşturduğu yüzeye ise pareto yüzeyi de denilmektedir.

DOI: 10.56554/jtom.1075286



Şekil 1. Etkin ve etkin olmayan noktaların gösterimi

Çalışmamızın ilerleyen kısımlarında çalışılacak olan yöntemler işte yukarıda tanımlanan pareto yüzeyi ya da çözüm kümesini bulmaya çalışacaktır. Kesin metotlar bu çözüm kümesinin tamamının bulunmasını garanti ederken, sezgisel metotlar bu kümenin tamamını bulmayı garanti etmemektedir.
Yukarıda yer alan örneğimizde, literatürde en sık kullanılan amaç ikilisi olan mesafe/süre ikilisi kullanılmıştır. Bu örnek haricinde iki amacın bir biri ile daha çok çeliştiği pek çok problem ele alınabilir. Örneğin; paralı otoyol ve köprüler ile ücretsiz yolların yer aldığı bir güzergahta süre ve maliyet amaçları çelişecektir. Keza ucuz taşıma yolları (örneğin deniz), orta maliyette taşıma yolları (kara ve demiryolu) ya da pahalı taşıma yollarının (havayolu) kullanılabildiği bir taşımacılık probleminde yine süre ve maliyet amaçları çelişecektir. Bir öğrencinin eğlenceye ayıracağı zaman ile derslerden alacağı not ortalaması çelişecektir, bu durumda yüksek not almak için eğlenceye daha az zaman ayırabilecek ya da daha yüksek not almak için eğlenceye daha az zaman ayırabilecektir. Bir elektronik devre tasarımında daha hızlı ya da daha yüksek kapasiteli devre elemanları kullanılması maliyetleri artıracaktır. Bu durumda yine devrenin hız/kapasitesi ile maliyetler çelişecektir. Bu örneklerin tamamında bir pareto optimal çözüm kümesi ve pareto optimal çözüm yüzeyi oluşacaktır. Bu çözümlerin her birisi etkindir ve dolayısıyla hiç biri diğerini domine etmemektedir. Görülebileceği üzere iki amaçlı en kısa yol probleminin gerçek hayatta bir çok örneği bulunabilmektedir.
Şu ana kadar iki amaçlı en kısa yol problemlerinin en genel hali olan en küçük toplam-en küçük toplam problemi anlatılmıştır. En küçük toplam-en küçük toplam probleminde iki amacın de en kısa olması hedeflenmiştir. Bunun haricinde literatürde en küçük toplam-limit ve en küçük toplam-en küçük azami değer tarzı iki amaçlı problemler de ortaya konulmuştur. Bunlardan en küçük toplam-limit türü problemlere örnek olarak, belli bir yoldan geçebilecek ağırlığın sınırlı olmasına ilişkin problemler örnek olarak verilebilir. Bu problemlerde güzergah üzerindeki yollardan geçebilecek araçların ağırlık sınırları mevcuttur. Bu tür problemlere verilebilecek diğer bir örnek ise belli noktalardan geçebilecek tehlikeli madde türünün sınırlı olmasına ilişkin problemlerdir (Erkut,1998).Bu problemlerde belli maddeleri taşıyan araçların, güzergahtaki bazı yolları kullanmasına ya da şehirlerden geçmelerine izin verilmez. En küçük toplam-en küçük azami değer türü problemlere örnek olaraksa belli bir yolda gidilen en yüksek eğimlerin minimize edilmesi örnek verilebilir. Bu iki tür problemde ikinci amacın toplamı değil, güzergah içinde ulaştığı değerler dikkate alınmaktadır. Bu iki tür problem çalışmamızın konusu dışında kalmaktadır.

3. Problemin çözüm yöntemleri

Çalışmamızın birinci bölümünde, literatürde çok amaçlı en kısa yol problemine ilişkin yapılan çalışmalarda ulaşılan yöntemlerin kesin yöntemler ve sezgiseller olarak ikiye ayrıldığı belirtilmişti. Kesin yöntemler, problemin çözümü olan pareto optimal noktaların tamamına ulaşan yöntemlerdir. Sezgiseller ise daha hızlı algoritmalar kullanmakta fakat pareto optimal noktaların tamamına ulaşma garantisi vermemektedirler. Bu bölümde bu yöntemler söz konusu ayrıma göre açıklanacaktır.

3.1 Kesin Yöntemler

Yukarıda da belirtildiği üzere kesin yöntemler etiketleme yöntemleri ve ağaç/güzergah yöntemleri olarak ikiye ayrılmaktadır.

3.1.1 Etiketleme Yöntemleri:

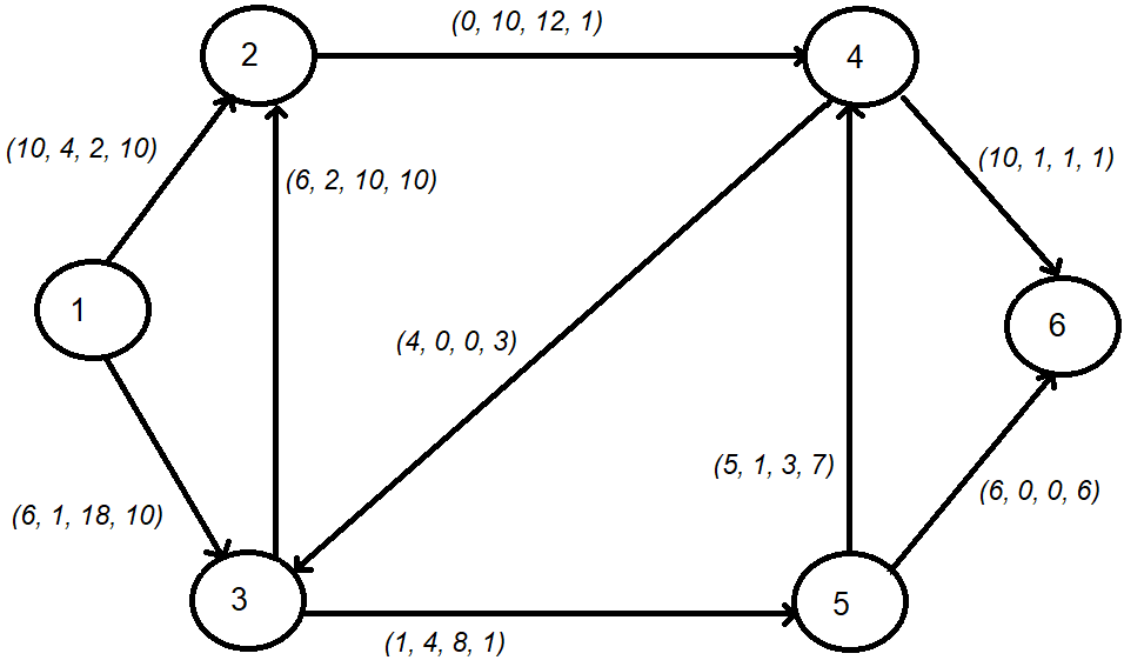
Etiketleme yöntemleri etiket belirleme (label setting) ve etiket düzeltme (label correction) olarak ikiye ayrılmaktadır.

DOI: 10.56554/jtom.1075286

3.1.1.1 Etiket belirleme

Etiket belirleme yöntemleri Martins (1984) algoritmasına dayanmaktadır. Aşağıda bu yöntem adımları ile anlatılacaktır.

Bu algoritmada her aşamada bulunulan konumu özetleyen etiketler üretilmektedir. İki amaçlı bir problem için üretilen etiket şu şekilde olmaktadır: <W1,W2,(P, P_{no}) >. Etiketin içerisinde gösterilen W1 ilk amaç açısından W2 ise ikinci amaç açısından başlangıç düğümünden gelinen düğüme kadar ulaşılan toplam ağırlıkları göstermektedir. P, ilgili düğüme gelinen düğümün numarasını, P_{no} ise önceki düğümün kaçıncı etiketi temel alınarak şu andaki etiketin oluşturulduğunu göstermektedir. Örneğin, <15,7,(4,12)> şeklindeki bir etiket; 4'üncü düğümün 12'nci etiketi temel alınarak üretilmiş olan, ilk amaç açısından 15 ikinci amaç açısından ise 7 ağırlığa sahip bir güzergahı gösteren etiket olarak okunmalıdır. Başlangıç düğümünde tek etiket bulunmaktadır bu etiket ise <0,0,(-,-)> şeklinde olmaktadır. Bu bilgilere göre Martins'in (1984) çalışmasında yer alan ve birçok çalışmada kullanılan örneği aşağıda açıklayacağız. Şekil-2 tarafımızca yeniden çizilmiştir, ancak içerik olarak Martins'in (1984) çalışmasındaki şeklin aynısıdır.



Şekil 2. Dört kriterli çizge

Kaynak: Martins :1984

Yukarıda yer alan Şekil-2 üzerinden etiket seçimi algoritması açıklanacaktır. Algoritmanın basamakları şu şekildedir (Martins , 1984):

Basamak 0: Başlangıç düğümüne geçici etiket olarak <0,0,0,0,(-,-)> atanacaktır.

Basamak 1: a- Eğer geçici düğüm kümesi boş ise 3 nolu basamağa git. Boş değilse, geçici etiketler kümesinden alfabetik olarak en küçük düğümü seç bu etiketi kalıcı etiketler kümesine at. Bu noktada alfabetik olarak (leksikografik) en küçük düğüm şu şekilde belirlenecektir. İlk amaç açısından en küçük olan alfabetik olarak en küçüktür, eğer bu şekilde birden fazla etiket varsa, ikinci amaç ağırlığı en küçük olan etiket en küçüktür, bu şekilde birden fazla etiket varsa bu durumda bir sonraki ağırlığa bakılır, aynı şekilde son ağırlığa kadar gidilir. Eğer bütün ağırlıklar eşitse bu etiketlerden herhangi biri alfabetik olarak en küçük olarak kabul edilir.

b-Alfabetik olarak en küçük geçici etiket, kalıcı etiket olarak etiketlendikten sonra ve kalıcı etiketler kümesine alındıktan sonra, bu etiket kullanılarak geçici etiketler kümesine yeni etiketler eklenecektir. Burada, örnek olarak kalıcı etiket olarak kullanılan etiketin i'nci düğüme ait olduğunu varsayalım. Bu düğümden çıkan bütün kenarlar için birer etiket oluşturulacaktır. Bu geçici etiketler oluşturulurken, şöyle bir hesaplama yapılacaktır. Kalıcı etiketimizdeki amaç ağırlıkları sırasıyla <c_1, c_2,..,c_n> olsun. Bu düğümden çıkan herhangi bir kenarın ağırlıklarının <w_1,w_2,...w_n> olduğunu varsayalım. Bu durumda yeni oluşturulacak etiketin ağırlıkları sırasıyla <c_1+w_1, c_2+w_2, ...,c_n+w_n> olacaktır. Bu aşamada başlangıç düğümünden yeni geçici etiketin oluşturulduğu düğüme gelen bütün geçici etiketler tek tek kontrol edilecek ve domine edilmiş olanlar geçici etiket listesinden silinecektir.

Basamak 2: Basamak 1'e dön.

Basamak 3: Başlangıç düğümünden başlayıp, bitiş düğümüne giden bütün domine edilmemiş etiketleri tespit et.

DOI: 10.56554/jtom.1075286

Basamak 4: Algoritmayı durdur.

Yukarıdaki şekilde gösterilen harita üzerinde bu algoritmayı uygulayalım. Aşağıda yer alan Tablo-4, Martins'in (1984) çalışmasında yer alan tablodan alınmıştır. Gölgelenmiş hücreler kalıcı etiketleri, açık renkliler ise geçici etiketleri göstermektedir. Örneğin 4'üncü düğümün ikinci etiketi domine edilmemiştir, zira ilgili düğüme ilişkin olarak oluşturulan hiç bir etiket bütün amaçlarda daha düşük ağırlıklara sahip değildir. Aynı düğümün üçüncü etiketi ise domine edilmiştir, zira ilk etiket her amaç için daha düşük ağırlıklara sahiptir. Aynı düğümün iki ve üçüncü etiketleri ise birbirlerini domine etmemektedirler.

Tablo 4. Martins Algoritması Çözüm Süreci

Düğüm	Etiket 1	Etiket 2	Etiket 3	Etiket 4
1	<0,0,0,0,(-,-)>			
2	<10,4,2,10,(1,1)>	<12,3,28,20,(3,1)>	<20,16,24,24,(3,2)>	
3	<6,1,18,10,(1,1)>	<14,14,14,14,(4,2)>	<16,6,29,21,(4,1)>	
4	<12,6,29,18,(5,1)>	<10,14,14,11,(2,1)>	<12,13,40,21,(2,2)>	<20,19,25,22,(5,2)>
5	<7,5,26,11,(3,1)>	<15,18,22,15,(3,2)>		
6	<13,5,26,17,(5,1)>	<20,15,15,12,(4,2)>	<22,7,30,19,(4,1)>	<21,18,22,21,(5,2)>

Algoritma ilk çalıştığında öncelikle 1'nci düğümün ilk etiketi oluşturulacaktır. Bu etiket kullanılarak, 1'nci düğümden çıkan iki kenar için, 2 ve 3'üncü düğümlerin ilk etiketleri oluşturulacaktır. 1'nci düğümün ilk etiketi kalıcı etiketler arasına alınacaktır. Bunlar arasında alfabetik olarak en küçük olan 3'üncü düğümün ilk etiketi kullanılarak, bu düğümden çıkan iki kenar için 2'nci düğümün ikinci etiketi ile 5'nci düğümün ilk etiketi hazırlanacaktır. 3'üncü düğümün ilk etiketi kalıcı düğümler arasına alınacaktır. Bu şekilde devam edilerek, bitiş düğümü olan 6'ncı düğümün etiketleri hazırlanacaktır. En sonra hiç geçici etiket kalmadığında algoritma bitecek ve yukarıdaki tablo oluşacaktır. Burada bitiş düğümüne ait iki kalıcı etiket ise domine edilmemiş iki adet güzergahı gösterecektir. Bu güzergahlar belirlenirken, geriye doğru iz takibi (back tracking) algoritması kullanılacaktır. Örneğin 6'ncı düğümün ilk etiketine bakıldığında 5'nci düğümün ilk etiketinden oluşturulduğu görülecektir. 5'nci düğümün ilk etiketi 3'üncü düğümün ilk etiketinden, bu ise başlangıç düğümünün ilk etiketinden oluşturulmuştur. Bu durumda güzergah 1=> 3=> 5=> 6 şeklinde olacaktır. Bu güzergahın toplam amaç ağırlıkları ise <13,5,26,17> şeklinde olacaktır. Tek tek ilgili üç kenara (1=>3), (3=>5) ve (5=>6) ait amaç ağırlıkları toplandığında bu sonucun doğru olduğu görülecektir.

3.1.1.2 Etiket düzeltme

Etiket düzeltme yöntemlerinde ise Skriver'in (2000b) çalışması en başarılı sonuçları verdiği için bu yöntem üzerinden anlatılacaktır. Bu yöntem zaten etiket bazlı yöntemlerin de en hızlısıdır. Bu nedenle çalışmamızda yer alan analiz kısmında bu algoritma kullanılmaktadır.

Etiket düzeltme algoritmaları, Skriver (2000b) öncesinde de çalışılmış algoritmalardandır. Brumbaugh ve Smith (1989) tarafından yapılan uygulaması literatürde en çok bilinenlerdendir. Skriver ise bu algoritma üzerinde iyileştirmeler yapmış ve kendi algoritmasını oluşturmuştur. Bu nedenle öncelikle Brumbaugh-Smith algoritmasını aşağıda açıklayacağız.

Bu algoritmanın etiketleri hazırlama kısmı, etiket düzenleme algoritmasına benzemektedir. Ancak burada seçimler sırasında etiketler üzerinden değil, düğümler üzerinden gidilmektedir. İncelenmesi gereken düğümler kümesi boş kalana kadar, sırasıyla bu kümeden bir adet düğüm alınmakta, bu düğümden çıkan kenarlar teker teker incelenmektedir. Bu inceleme sonucunda ortaya çıkarılan yeni etiketler ile eski etiketler, etiket kümesinde birleştirilmektedir. Aynı etiketten birden fazla ortaya çıkmaması için de, aynı özelliklere sahip bir etiket varsa, birleştirmeye dahil edilmemektedir. İşte bu algoritmanın en büyük işlem yükünü bu birleştirme algoritması içermektedir.

Yukarıdaki algoritmada Skriver (2000b) tarafından yapılan iyileştirmeler, Tung ve Chew'in (1988) çalışmasında kullanılan ve ileride A* bazlı çalışmalarda da kullanılan, bulunulan düğümden varış düğümüne her amaç için en kısa yol uzunluklarının tespitini kullanan bir iyileştirmedir. Böylelikle bazı kenarlar hiç kümeye dahil edilmeyecektir. Skriver'ın (2000b) çalışmasında kullanılan iyileştirme algoritması şu şekildedir:

Bir i düğümünden, bir j düğümüne giden kenarın etiketleme işlemine katılıp katılmayacağını kontrol edilecektir. Burada i düğümü için $w_1(i)<w_2(i)<w_3(i)<....<w_k(i)$ ve $v_k(i)<...<v_3(i)<v_2(i)<v_1(i)$ şeklinde (w ilk amacın maliyeti, v ise ikinci amacın maliyeti olsun) bir etiket kümesi olduğunu varsayalım. j düğümü için ise $w_1(j)<w_2(j)<w_3(j)<....<w_q(j)$ ve $v_q(j)<...<v_3(j)<v_2(j)<v_1(j)$ şeklinde bir etiket kümesi olduğunu varsayalım. i ve j düğümleri arasında yer alan kenarın amaç ağırlıkları ise sırasıyla a ve b olsun. Bu durumda ortaya çıkacak durumları değerlendirelim.

Eğer $w_1(i)+a>w_q(j)$ ise bu durumda eğer $v_k(i)+b>v_q(j)$ ise, üretilecek her etiket domine edilecek, dolayısıyla birleşme işlemi herhangi bir değişikliğe neden olmayacaktır. Bu nedenle ilgili kenar etiketleme işlemine alınmayacaktır.

DOI: 10.56554/jtom.1075286

Benzer şekilde, eğer $v_k(i)+b>v_1(j)$ ise bu durumda eğer $w_1(i)+a>w_1(j)$ ise, üretilecek her etiket domine edilecek, dolayısıyla birleşme işlemi herhangi bir değişikliğe neden olmayacaktır. Bu nedenle ilgili kenar etiketleme işlemine alınmayacaktır.
Aksi durumlarda ise birleşme işlemi yapılabilecektir. Bu birleşme işleminin sözde kodu Tablo 5'te gösterildiği şekildedir. (Skriver,2000b)

Tablo 5. Etiket Düzeltme Algoritması Birleşme Sözde Kodu

Eğer $w_1(i)+a< (w_q(j)$ ise bir şey yapma; (çünkü (i,j) düğümü gelecek vaadediyor)
değilse (yani >= ise)
Eğer $v_k(i)+b<v_q(j)$ ise bir şey yapma; ((çünkü (i,j) düğümü gelecek vaadediyor)
değilse j'yi i'nin çıkışlarından çıkar (çünkü (i,j) düğümü domine edilmiştir)
Eğer $v_k(i)+b<v_1(j)$ ise bir şey yapma; (çünkü (i,j) düğümü gelecek vaadediyor)
değilse (yani >= ise)
Eğer $w_1(i)+a<w_1(j)$ ise bir şey yapma; ((çünkü (i,j) düğümü gelecek vaadediyor)
değilse j'yi i'nin çıkışlarından çıkar (çünkü (i,j) düğümü domine edilmiştir).

Kaynak: Skriver : 2000b

Bu şekilde oluşturulan etiketlerden varış noktası için oluşturulan domine edilmemiş etiketler, çözüm kümesini oluşturacaktır.

3.1.2 Ağaç/Güzergah yöntemleri

Ağaç/güzergah yöntemleri ise iki aşamalı çözümler ve K adet en kısa yol algoritması olarak ikiye ayrılmaktadır.

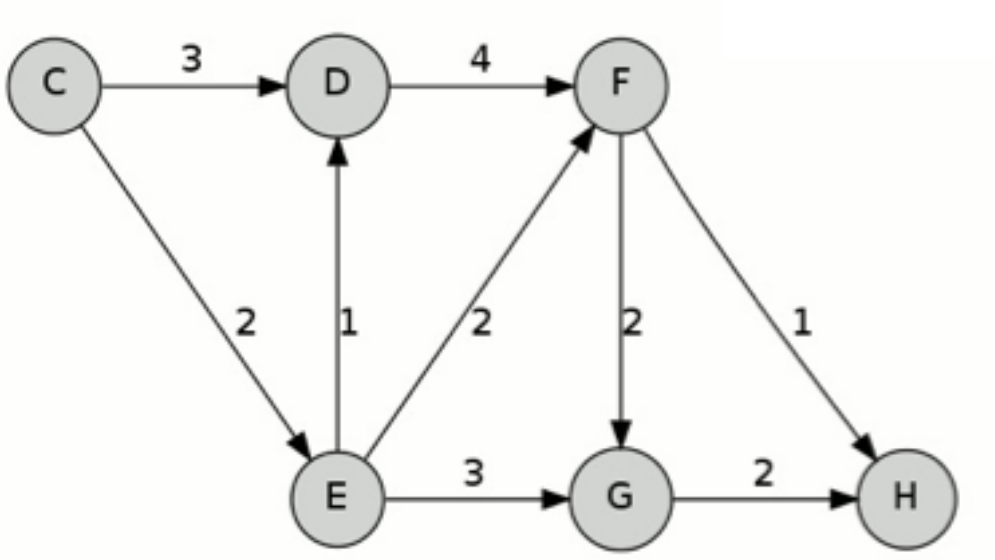
3.1.2.1 İki aşamalı çözümler:

İki aşamalı çözümler, ağaç araması algoritmaları kullanılarak yapılan algoritmalardır. Bu algoritmalarda, iki amaç açısından en kısa yollar bulunduktan sonra, bu yollarda ulaşılan değerlerin ara değerleri aranmaktadır. Literatürde çok fazla uygulaması bulunmamaktadır. Örnek olarak Murthy ve Olson'un (1991) çalışması gösterilebilir. Bu çalışmaya ilişkin yapılan karşılaştırmalarda, bu algoritmanın ilk aşamasının bile etiket düzeltme algoritmalarından yavaş olduğu sonucuna varılmıştır (Skriver,2000a). Bu algoritmanın diğer algoritmalar ile kıyaslandığı diğer bir çalışma da (Raith ve Ehhrgott., 2009), bu yöntemi etiket belirleme ve etiket düzeltme yöntemlerine göre daha yavaş olarak değerlendirmiştir.

3.1.2.2 K adet en kısa yol algoritması

Bu algoritma Yen'in (1971) tek amaçlı en kısa K adet yola ulaşma probleminin üzerine bina edilmiştir. Söz konusu problemde, sadece en kısa yol değil, bulunan yollar uzunluklarına göre sıralandığında en kısadan başlamak suretiyle elde edilen K adet yolun sıralanmasına dayanmaktadır.
K adet en kısa yol bulunurken iki aşamalı bir algoritma kullanılır. İlk aşamasında en kısa yol bulunur. Bu en kısa yol, en kısa yollar kümesine eklenir. İkinci aşamasında ise ara bir noktada olunduğu varsayılır.
Bu noktada daha önceden n-1 adet en kısa yol olduğu varsayılarak, n'nci ($n<=K$) en kısa yol aranmaktadır. Bu güzergah belirlenirken daha önce ortaya çıkan n-1 adet en kısa yol ele alınır. Bu güzergahlarda geçilen kenarların ağırlıkları, sırasıyla sonsuz yapılır ve en kısa yol algoritması uygulanır. Bu şekilde elde edilen yeni güzergahlar bir kümeye atılır, bunların en kısa olanı n'nci en kısa yol olarak en kısa yollar kümesine alınır. (Yen, 1971)
Bu algoritmayı bir örnek üzerinden göstermek faydalı olacaktır. Şekil-3'te görülen örnek Wikipedia (2021)'den alınmıştır.

DOI: 10.56554/jtom.1075286



Şekil 3. K en kısa yol için çizge

Kaynak: Wikipedia (2021)

Bu örnekte C düğümünden H düğümüne en kısa 3 güzergâh bulunacaktır. Açıktır ki, herhangi bir en kısa yol algoritması ile C-E-F-H güzergahı en kısa yol olarak tespit edilebilir. Bu güzergahın uzunluğu ise 2+2+1 = 5 birim olacaktır.

İkinci en kısa yol bulunurken, sırasıyla yukarıdaki güzergahta kullanılan kenarların (C-E,E-F, F-H) değerleri sonsuza eşitlenecek (böylelikle kaldırılacak) ve başlangıçtan ilgili kenarın başlangıç düğümüne ardından da o düğümden bitişe birer kere en kısa yol algoritması çalıştırılacak ve bunların toplamı aday güzergah kümesine atılacaktır. Burada önce C-E kenarı çıkarılacak, böylelikle bu kenar kullanılmadan C'den bitişe en kısa yol olan C-D-F-H güzergahı bulunacaktır. Başlangıç zaten C olduğundan bu güzergah 8 birim uzunluğundaki C-D-F-H olacaktır. Bu güzergah ikinci en kısa yol aday kümesine atılacaktır. Ardından E-F kenarı çıkarılacaktır. Bu kenarın başlangıç düğümü olan E'den H'ye en kısa yol E-G-H olacaktır. Böylelikle ikinci en kısa yol aday kümesine 7 birimlik C-E-G-H güzergahı atılacaktır. Son olarak F-H kenarı çıkarılacaktır. F'den bitişe en kısa yol F-G-H güzergahıdır. Böylelikle 8 birimlik C-E-F-G-H güzergahı, en kısa ikinci yol aday kümesine atılacaktır. Bu kümedeki en kısa güzergah olan C-E-G-H, ikinci en kısa güzergah olacaktır. İkinci aşamanın sonunda en kısa yol C-E-F-H, en kısa ikinci yol ise C-E-G-H olarak tespit edilmiştir.

Üçüncü en kısa yolu tespit ederken, bu sefer ikinci en kısa yol ele alınacaktır. İkinci en kısa yol aday kümesinde kalan yollar, üçüncü en kısa yol aday kümesine otomatik olarak alınacaktır. İkinci en kısa yolun ilk kenarı C-E, çıkarılacaktır. Bu durumda C'den H'ye en kısa yol 8 birimlik C-D-F-H olacaktır. Bu güzergah zaten en kısa yol aday kümesinde olduğundan bu artık eklenmeyecektir. Ardından E-G kenarı çıkarılacaktır. Bu durumda E'den H'ye en kısa yol C-E-F-H olacaktır, bu ise en kısa yol kümesinde yer aldığından tekrar eklenmeyecektir. Son olarak G-H kenarı çıkarıldığında ise en kısa yol oluşmayacaktır. Bu durumda en kısa yol aday kümesindeki en kısa yol olan C-D-F-H üçüncü en kısa yol olacaktır.

Bu problem her ne kadar tek amaçlı bir en kısa yol problemi olsa da, çözüm tek bir güzergah değil de birden fazla güzergahtan oluşan bir çözüm kümesi olduğu için, bir takım değişiklikler ile çift amaçlı en kısa yol probleminin çözümünde kullanılacak hale gelmektedir. Burada ki mantık K sayısının sabit tutulmamasına dayanmaktadır. Öncelikle iki amaç açısından da en kısa yollar bulunmaktadır. Ardından, amaçlardan birisi kullanılarak, diğer amaç için en kısa yol elde edilene kadar bu algoritma çalıştırılmaktadır. Algoritma bulunan güzergahları sırası ile bulduğu için, diğer amaç açısından en kısa yol bulunduğu anda, iki amaçlı en kısa yol problemi için bütün çözüm kümesi elde edilmektedir. (Climaco ve Martins,1982)

3.2 Sezgiseller

Literatür taraması kısmında da belirttiğimiz üzere, problemde yoğun olarak A* temelli algoritmalar çalışılmış, bunun yanında damlacık, genetik tabanlı sezgiseller de kullanılmıştır. Literatürde en çok A* temelli algoritmalar çalışıldığı için aşağıda bunların en bilinenlerine yer verilmiştir.

3.2.1 A* Temelli Algoritmalar

A* en kısa yol algoritmaları için kullanılan bir sezgisel paradigmasıdır. Hart et al tarafından (Hart, Nillson ve Raphael, 1968) geliştirilmiş olan bu paradigma aslında doğrudan bir çözüm yöntemi sunmak yerine bir çözüm

DOI: 10.56554/jtom.1075286

yönteminin sahip olması gereken özellikleri belirlemiştir. Bu paradigmaya göre, en kısa yol tespiti için ilerlenecek düğüm seçilirken bir A* fonksiyon değeri tespit edilmeli, eldeki aday düğümlerden en küçük A* fonksiyon değerine sahip olanına ilerlenmelidir. Bu A* fonksiyonu ise (örneğin F(N), N burada bir düğümü ifade etmektedir.), başlangıçtan ilgili düğüme gelene kadar gerçekleşen yol maliyeti (Bu C(N) olsun) ile ilgili düğümden sonuç düğümüne gidişe ilişkin sezgisel fonksiyon (H(N) olsun) değeri toplamı olacaktır. Yani ilerlenecek düğüm seçilirken F(N)=C(N) + H(N) değeri en düşük olan düğüme ilerlenecektir. C(N) gerçekleşmiş bir değeri, H(N) ise sezgisel ile bulunan bir değeri ifade etmektedir. Eğer H(N) tutarlı bir fonksiyon ise, A* algoritması kesin çözüm sunacaktır. (Hart et al., 1968).

3.2.1.1 MOA*

Bu algoritma, çok amaçlı A* (multi objective A*) olarak isimlendirilmiştir. A* tabanlı çok amaçlı en kısa yol algoritmalarının, bilgimiz dahilinde, ilkidir. Stewart ve White tarafından (1991) geliştirilmiştir. NAMOA* algoritması temel olarak bu algoritmayı almaktadır. Ancak aralarında küçük farklılıklar vardır. NAMOA* algoritması yapılan çalışmalarda performans olarak MOA*'dan daha iyi sonuçlar verdiği için aşağıda bu algoritma daha detaylı olarak açıklanacaktır.

3.2.1.2 NAMOA* ve NAMOA*dr

Bu algoritmalar ise Mandow et al (2010) tarafından geliştirilmiştir. Yine Mandow et al (2015) tarafından algoritmayı hızlandırılacak geliştirmeler yapılmıştır. İkinci algoritma, ilk algoritmanın boyutlarını küçültmek suretiyle hızlandırılmış şeklidir. NAMOA* algoritması temel olarak MOA* algoritmasını almaktadır, ancak MOA* düğüm seçme temelinde ilerlerken, NAMOA* etiket seçme temelinde ilerlemektedir.

NAMOA* algoritması da aslında temelde bir etiket seçme algoritmasıdır. Martins (1984) algoritmasında olduğu gibi herhangi bir etiket işlenirken, o etiketin sahibi olan düğümden gidilen her düğüm için yeni bir etiket düzenlenmektedir. Ancak burada geçici etiket, kalıcı etiket ayrımı yapılmamaktadır. Her etiket düzenlendiğinde, bu etiketler bir kümenin içine atılmakta, bu kümedeki etiketlerden, A* fonksiyon değerleri en küçük olan etiket seçilerek devam edilmektedir. Bu şekilde sonuç düğümüne varıldığında, oluşturulan etiket, sonuç düğümü için çözüm kümesinin elemanlarından birisi olmaktadır. Doğal olarak her etiketten yeni etiket üretilmeyecektir, eğer bir etiket, bulunduğu düğümde daha önceden oluşturulan etiketler tarafından domine edilmekteyse, artık o etiket işlenmeyecektir (Mandow et al,2010).

Bu algoritmanın hızlı çalışabilmesi için etiket kümesinin içindeki en düşük alfabetik A* değerine sahip etiketin seçilmesinden yığın ağacı (heap tree) veri yapısı kullanılmalıdır. Aksi taktirde orta boyutlu bir çizgede, etiket kümesinin büyüklüğü 10.000'lerle ifade edilebilecek boyuta kısa sürede gelmektedir. Bu kümenin en küçüğünü seçmek için yığın ağacı kullanılması durumunda toplam işlem sayısı 15-20 adet civarında olacakken (O(logN)), normal seçme yöntemleri ile işlem sayısı da küme büyüklüğü ile orantılı olacaktır (O(N)). NAMOA* algoritması sezgisel olarak ilgili düğümden sonuç düğümüne amaçların her birisi için en kısa yol değerini kullanmaktadır (Mandow et al, 2010).

Bu algoritmanın sözde kodu aşağıda Tablo-6'da görüldüğü gibidir. Bu sözde kodda S, düğümlerin, E kenarların oluşturduğu küme, c ise kaç tane amaç varsa o kadar ağırlıktan oluşan ve kenarların ağırlıklarını içeren kümedir. G_{op} kümesi her hangi bir düğüm için açık olan etiketlerin tutulduğu küme, G_{cl} kümesi ise her hangi bir düğüm için kapalı olan etiketlerin tutulduğu kümedir. Açık etiketler, yeni etiket üretimi için açık olan etiketlerdir, kapalı etiketler ise artık etiket üretimine kapatılmış etiketlerden oluşmaktadır. Bu kümede sonuç düğümü için en son kalan etiketler pareto optimal çözüm kümesini verecektir. Aşağıdaki iki algoritma da A* tabanlı olduğu için bir sezgisel fonksiyon (h*) ve o ana kadarki mesafeyi tutan c fonksiyonundan oluşan bir f* fonksiyonu mevcut olacaktır. Bu fonksiyonlar çalışmamızın A* kısmında açıklanmıştır.

Tablo 6. NAMOA* sözde kodu

Girdi : Bir arama problemi (S, E, c, s_{bas} , s_{bit}) ve tutarlı bir h* sezgisel fonksiyonu
Çıktı: Pareto optimal çözüm kümesi
SOL←∅
her s∈ S için yap
$G_{op}(s) \leftarrow \emptyset$; $G_{cl}(s) \leftarrow \emptyset$ (başlangıç atamaları)
$G_{op}(s) \leftarrow \{(0,0)\}$ ata((0, 0)) ← ∅ (herhangi bir etiket için ata düğümünü tutacaktır) Open kümesini oluştur ve başlangıç etiketini, (s_{bas} , (0, 0), h(s_{bas})) ekle
Open≠ ∅ sürece yap
Open kümesi içindeki alfabetik (lexicographic) olarak en küçük f* değerine sahip etiketi al
Bu etiketi G_{op} kümesinden çıkar ve G_{cl} kümesine ekle
eğer s = s_{bit} ise
Bu etiketi SOL kümesine ekle ve Open kümesi içinden f değeri bu f* değeri tarafından domine edilmiş bütün etiketleri çıkar.

DOI: 10.56554/jtom.1075286

Devam et
her t ∈ komşular(s) için yap
$g_t \leftarrow g_s + c(s,t)$ (s düğümünden komşuluğunda olan diğer düğümlere etiket oluştur) Eğer bu etiket domine edilmemiş bir etiket ise:
g_t etiketin g_s etiketini ata olarak ekle
Devam et
Eğer oluşan etiket domine edilmiş bir etiket ise etiketi yoksay ve sonraki komşuya geç
Aksi halde, bu yeni etiket tarafından domine edilmiş olan bütün etiketleri open ve closed kümelerinden çıkar ve bu etiketi Open kümesine ekle, bu etiketin atası olarak g_s etiketini ekle
SOL kümesini çözüm olarak sun

Kaynak: (Ulloa et al. 2020)

3.2.1.3 BOA*

Bu algoritma ise Ulloa et al (2020) tarafından geliştirilmiştir. Sadece 2 amaçlı olması durumunda çalışmaktadır. Ancak problemin boyutunu oldukça küçülttüğünden, çok hızlı sonuç vermektedir. Bu algoritmada NAMOA* algoritmasına göre yapılan geliştirme sayesinde problem neredeyse tek amaçlı bir en kısa yol algoritması seviyesine gelmiştir. Algoritma A* temelli olduğu için bir A* fonksiyonu kullanmaktadır, sezgisel olarak da aynen NAMOA*'ın kullandığı gibi ilgili düğümden sonuç düğümüne olan en kısa yol değerlerini sezgisel fonksiyon değeri olarak kullanmaktadır. Etiket kümesinin en küçük değerli etiketini seçmek için de aynen NAMOA* gibi yığın ağacı veri yapısını kullanmaktadır. Ancak herhangi bir etiketin domine edilip edilmediğinin kontrolü sırasında, ilgili düğüm için önceden oluşan bütün etiketleri kontrol etmek yerine, ilgili düğüm için başlangıçtan ilgili düğüme ikinci amaç açısından daha önceden ulaşılan en kısa mesafe değerini kontrol etmekle yetinmektedir (Ulloa et al 2020). Zira, etiketler seçilirken ilk amaç açısından en küçüğünden başlanarak ilerlenmektedir. Dolayısıyla, herhangi bir etiket daha önceden işlendiyse, mutlaka daha küçük ilk amaç değerine sahip olmalıdır. Bu durumda şu anda işlenen etiket, daha önceden ulaşılan en küçük ikinci amaç değerinden daha küçük değilse, zaten domine edilmiş bir etiket olacaktır. Bu sayede bütün etiketlerin ikinci amacı ile karşılaştırmak yerine, söz konusu etiket için daha önce ulaşılan en küçük ikinci amaç değeri ile karşılaştırılarak kontrol süresi sabitlenmektedir. Bu da algoritmayı çok hızlı hale getirmektedir.
BOA* algoritmasının sözde kodu aşağıda Tablo-7'de görüldüğü gibidir.

Tablo 7. BOA* algoritması sözde kodu

Girdi : Bir arama problemi (S, E, c, s_{bas} , s_{bit}) ve tutarlı bir h* sezgisel fonksiyonu
Çıktı: Pareto optimal çözüm kümesi
SOL←∅
her s∈ S için yap
$g_2^{min}(s) \leftarrow \infty$ (her düğümün g_2 değerini sonsuza eşitle)
Başlangıç düğümü için atamaları yap: etiket oluştur, c fonksiyon değerini sıfıra eşitle, ebeveyn kümesinde ebeveyn kümesini boş küme yap.
Open kümesini oluştur ve başlangıç düğümü için oluşturulan etiketi ekle
Open≠ ∅ sürece yap
Open kümesi içindeki alfabetik (lexicographic) olarak en küçük f* değerine sahip etiketi al
Eğer ilgili etiketin g_2 değeri, o düğüm için oluşan en küçük g_2 değerinden büyükse ya da o etiketin f_2 değeri, o düğüm için oluşan en küçük g_2 değerinden büyükse bu etiketi atla, bir sonraki etikete geç
Değilse o düğüm için $g_2^{min}(s)$ değerini bu etiketin g_2 değeri olarak yenile.
Eğer etiketin ulaştığı düğüm bitiş düğümü ise
Bu etiketi çözüm kümesine ekle ve bir sonraki etikete geç
her t ∈ komşular(s) için yap
t için yeni etiket oluştur. Bu etiket için f ve g değerlerini hesapla. Eğer ilgili etiketin g_2 değeri, o düğüm için oluşan en küçük g_2 değerinden büyükse ya da o etiketin f_2 değeri, o düğüm için oluşan en küçük g_2 değerinden büyükse bu etiketi atla, bir sonraki etikete geç
Değilse ilgili etiketi Open kümesine ekle
SOL kümesini çözüm olarak sun

Kaynak: (Ulloa v.d. 2020)

3.2.2 Diğer Sezgiseller

DOI: 10.56554/jtom.1075286

Çalışmamızın 1'nci bölümünde de bahsedildiği üzere, literatürde temel olarak A* tabanlı algoritmalar çalışılmış olsa da damlacık algoritması ve genetik algoritma tabanlı çalışmalar da mevcuttur. Ancak bu çalışmaların A* tabanlı sezgisellere göre yavaş kaldığı çeşitli çalışmalarda gösterilmiştir. Bu nedenle bu çalışmada ki uygulamalarımızda A* tabanlı sezgisellere yer verilmiştir.

4. Uygulama

Çalışmamızın bu bölümü iki kısımdan oluşmaktadır. İlk kısımda önceki bölümde açıklanan yöntemlerden etiket düzeltme, NAMOA* ve BOA* algoritmaları çeşitli veri setleri kullanılarak karşılaştırılacaktır. İkinci kısımda ise literatürde yapılan benzer karşılaştırmalar ele alınacaktır.

4.1 Veri Setleri Kullanılarak Karşılaştırma:

Çalışmamızda temel olarak iki farklı kaynaktan alınan veri seti kullanılacaktır. İlk kaynak, Princeton Üniversitesine (Princeton) ait açık kaynaklarda yer alan 4 farklı büyüklükteki tek amaçlı en kısa yol verisi temel alınarak hazırlanan çeşitli haritalar kullanılacaktır. Söz konusu haritalar Sedgewick ve Wayne'in (2014) kitabında da kullanılan haritalardır. Princeton üniversitesi açık kaynaklarında bulunan haritalar tek amaçlı en kısa yol haritalarıdır. Bu haritalara python programlama dili kullanılarak rastgele (randomize) ikinci amaç ağırlıkları eklenmiştir. Söz konusu işlem sonucunda haritalarda yer alan iki farklı amacın çelişip çelişmediğinin kontrolü amacıyla iki amaca ilişkin ağırlık değerlerinin korelasyonları alınmıştır. Bu korelasyon değerleri aşağıda Tablo-8'de gösterilmektedir.

Tablo 8. Amaç değişkenleri korelasyon tablosu

Veri Adı	Amaçlar arası korelasyon değeri
mediumEWD	-0.4783
1000EWD	-0.4413
10000EWD	-0.3876
Rome	-0.1105

Diğer veri kaynağı ise, ABD Ayrık Matematik ve Teorik Bilgisayar Bilimleri Enstitüsünün düzenlediği, ABD gerçek karayolu verileri üzerinde en kısa yol algoritmalarına ilişkin olarak düzenlenmiş olan (Dimacs,2005) yarışmada kullanılan çeşitli eyalet veya bölge haritalarıdır. Dimacs haritalarında ise Amerikan Nüfus İdaresi (Census Bureau) verileri esas alındığı için iki amaç (mesafe ve yolculuk süresi) de hazır olarak yer almaktadır. Bu haritalarda iki amaç, tek bir dosyada birleştirilmiştir.

Aşağıda yer alan Tablo 9'da çalışmamızda kullanılacak olan haritalardaki düğüm ve kenar sayıları yer almaktadır. Ayrıca kenar sayısının düğüm sayısına oranı da ortalama bir yoğunluk ölçütü olarak tabloya eklenmiştir.

Tablo 9. Çalışmada kullanılan haritalar

Veri Adı	**Düğüm Sayısı**	**Kenar Sayısı**	**Kenar Sayısı/ Düğüm Sayısı**
mediumEWD	250	2.546	10,18
1000EWD	1.000	16.866	16,87
10000EWD	10.000	123.462	12,35
Rome	3.353	8.870	2,65
Dimacs NewYork	264.346	733.846	2,78
Dimacs Bay Area	321.270	800.172	2,49
Dimacs Colorado	435.666	1.075.066	2,47
Dimacs Florida	1.070.376	2.712.798	2,53
Dimacs N.West US	1.207.945	2.840.208	2,35
Dimacs N.East US	1.524.453	3.897.636	2,56

Uygulamamızda, kesin algoritmaların en hızlısı olan etiket düzeltme algoritması kullanılmıştır. Sezgisellerden ise literatürde en çok çalışılan sınıf olan A* algoritmalarından ikisi (NAMOA* ve BOA*) kullanılmıştır. Böylece literatürde yer alan sezgiseller ile kesin yöntemlerin karşılaştırılması sağlanmış, hem süre karşılaştırması yapılmış hem de çözüm kümesinin tamamına ulaşılıp ulaşılamadığı kontrol edilmiştir.

Uygulamada verilerin seçiminde yığın ağacı veri yapısı kullanılmış, Intel i7-6700HQ 2.6 GHz işlemci, 16 GB Ram'e sahip bilgisayarlardan oluşan bir bilgisayar laboratuvarında aynı anda çalıştırılmıştır. Programlar Phyton programlama dili ile yazılmış, zaman ölçümleri, program tarafından otomatik olarak yapılmış ve çıktı dosyasına eklenmiştir.

Bu uygulama sonucunda, uygulamasını yapmış olduğumuz haritalar için elde ettiğimiz sonuçlar Tablo 10'da yer almaktadır. Tabloda her bir algoritma türü ve harita için çözüm süresi ile çözüm kümesi büyüklüğü yer almaktadır. Kesin çözüm algoritmalarının bu çözümlerin tamamına ulaşması beklenmektedir. Sezgisellerde ise kullanılan A*

DOI: 10.56554/jtom.1075286

sezgisel fonksiyonu, A* algoritmalarında "tutarlı" olarak belirtilen özellikleri taşıdığı için, bütün çözüm kümesinin bulunması beklenmektedir. Çözüm kümesi büyüklüğü sütununda işte bu çözüm kümesinde yer alan çözümlerin sayısı yer almaktadır. Aşağıda Tablo-10'dan da görülebileceği üzere, kesin yöntem olan etiket düzeltme yönteminde ulaşılan etiket adedine (domine edilmemiş çözüm sayısı) kadar diğer yöntemlerde de ulaşılmıştır.
Çözüm süresi olarak bir algoritmanın 70 saate kadar çalışmasına izin verilmiş, algoritmanın çözememesi durumunda, program durdurulmuş ve tablodaki ilgili yere not düşülmüştür.

Tablo 10. Çözüm özet tablosu

Harita Adı	Etiket Düzeltme		NAMOA*		BOA*	
	Süre	Çözüm Kümesi Büyüklüğü	Süre	Çözüm Kümesi Büyüklüğü	Süre	Çözüm Kümesi Büyüklüğü
mediumEWD	5.5 sn	19 etiket	0.4 sn	19 etiket	0.1 sn	19 etiket
1000EWD	3.990 sn	15 etiket	9.2 sn	15 etiket	0.4 sn	15 etiket
10000EWD	-	-	1.970 sn	311 etiket	65 sn	311 etiket
Rome	40 sn	27 etiket	1.5 sn	27 etiket	0.3 sn	27 etiket
Dimacs NewYork	-	-	127.000 sn	157 etiket	38 sn	157 etiket
Dimacs Bay Area	-	-	7.800 sn	58 etiket	19 sn	58 etiket
Dimacs Colorado	-	-	-	-	75 sn	166 etiket
Dimacs Florida	-	-	-	-	157 sn	63 etiket
Dimacs N.West US	-	-	-	-	5.500 sn	95 etiket
Dimacs N.East US	-	-	-	-	56.000 sn	677 etiket

-: 70 saatlik çalışmanın sonunda sonuca ulaşılamadığı için program kesilmiştir.

Yukarıdaki Tablo 10 sonuçlarından görülebileceği üzere, yöntemlerin tamamı çözüm buldukları haritalarda aynı sayıda etiket sayısına ulaşmıştır. Bu çözüm kümesinin tamamına ulaşıldığını göstermektedir. Süre açısından BOA* algoritması diğer iki yönteme göre her durumda daha hızlı çözüm elde etmiştir. NAMOA* algoritması ise her durumda etiket düzeltme algoritmasına göre hızlı çözüm elde etmiştir. Ancak BOA* algoritmasına göre, harita büyüdükçe yavaş kalmıştır. Bu bilgiler kapsamında iki amaçlı en kısa yol problemlerinde BOA* algoritmasının kullanılmasının uygun olduğu düşünülmektedir. Öte yandan amaç sayısı ikiden fazla olduğunda BOA* algoritması da kullanılamayacağı için, NAMOA* algoritmasının kullanılması uygun olacaktır.
Çözüm süreleri irdelendiğinde, kenar ve düğüm sayısı ile birlikte sürelerin arttığı görülmektedir. Ancak bazı haritalarda düğüm ve kenar sayıları daha az olmasına rağmen çözüm sürelerinin yüksek olduğu görülmektedir. Örneğin BOA* algoritması 10000EWD haritasını, NY haritasına ve Bay Area haritasına göre daha yavaş çözmüştür. Öte yandan hem NAMOA* hem BOA* algoritmaları Bay Area haritasını NY haritasına göre daha hızlı çözmüştür. Buradan çözüm sürelerinin sadece kenar/düğüm sayılarına göre değil aynı zamanda haritaların yapısına göre de değiştiği söylenebilir. Haritaların yapısı, karmaşıklığı ve bunun çözüm süresine etkisi ileride yapılacak çalışmalar için araştırma konusu olabilecektir.
Bu çalışmamızda çalışılan modellerde çift amaçlı en kısa yol problemlerinde en çok kullanılan en küçük toplam-en küçük toplam türü çözüm yöntemleri kullanılmıştır. Bu yöntemlerde iki amacın de en düşük değeri hedeflenmektedir. Bunun haricinde en küçük toplam-limit değer, ve en küçük toplam-en küçük azami değer türü problemler de mevcuttur. Ancak bu türlere ilişkin olarak bildiğimiz kadarıyla geliştirilmiş bir sezgisel mevcut değildir. Yani yukarıda anlatılan yöntemler ve yapılan uygulama, iki amaçlı en kısa yol problemlerinin en geneli için yapılmıştır. Öte yandan problemin daha özel durumları için de sezgiseller geliştirilebileceği gibi yukarıda sözü edilen sezgisellerin bu durumlara uyarlanması da ilerleyen çalışmalara konu olabilecektir.

4.2 Diğer Çalışmalarda Yapılan Uygulamalarla Karşılaştırma:

4.2.1 (Ulloa v.d., 2020) çalışması

Bu çalışma BOA* algoritmasının geliştirildiği makaledir. Bu makalede yapılan çalışmada iki ve çok amaçlı sezgisel metotlar karşılaştırılmıştır. Çalışmamızda kullandığımız BOA* ve NAMOA* algoritmaları söz konusu çalışmada karşılaştırılmıştır. Bu karşılaştırma sırasında, çalışmamızda kullanılan Dimacs NY, Dimacs Bay Area, Dimacs Colorado ve Dimacs Florida haritaları kullanılmıştır. Söz konusu çalışmada, bu çalışmaya benzer şekilde BOA* algoritmasının, NAMOA* algoritmasına göre çok daha hızlı çözüm bulduğu tespit edilmiştir.

4.2.2 (Pulido v.d., 2015) çalışması

Bu çalışmada NAMOA* algoritmasını geliştiren yazarlar tarafından, NAMOA* algoritmasından türetilen NAMOA* DR algoritması geliştirilmiştir. Bu çalışmada NAMOA* ve NAMOA* DR algoritmaları çalışmamızda kullanılan Dimacs NY haritası kullanılarak karşılaştırılmıştır. Çalışmamızda kullanılana benzer bir bilgisayar kullanılarak yapılan karşılaştırmada, NAMOA* algoritması, çalışmamıza elde edilen sürelere yakın sürelerde

DOI: 10.56554/jtom.1075286

sonuçlar vermiştir. Bu çalışmada kullanılan boyut küçültme (DR) metodolojisinin ileride yapılacak çalışmalarda kullanılması mümkündür.

4.2.3 (Skriver, 1999) çalışması

Bu çalışma nispeten eski bir çalışmadır. Çalışmanın önemi ise kesin yöntemler içinde en hızlısı olan etiket düzeltme yönteminin geliştirilmesidir. Bu çalışmada 100 ila 500 düğümlü haritalar kullanılmıştır. Çalışmanın yapıldığı dönemdeki bilgisayar imkanları değerlendirildiğinde, çalışmada elde edilen sürelerin makul olduğu görülecektir. Çalışmanın önemi ise bu çalışmamızda kullanılan kesin yöntemin belirlenmesinde kendisini göstermektedir. Zira etiket düzeltme yönteminin en hızlı kesin yöntem olduğu bu çalışma ile ortaya konulmaktadır.

4.2.4 BOA* üzerine yapılan çalışmalar (Ahmadi, Guido, Harabor, Philip. ,2021), (Goldin ve Salzman, 2021):

Bu çalışmalar BOA* algoritması üzerinde minör geliştirmeler yapmak üzere hazırlanmıştır. Bu çalışmalarda yine DIMACS haritaları kullanılmıştır. Süreler olarak BOA* algoritmasına benzer sürelere ulaşılmıştır. Bu çalışmalarda BOA* algoritmasında geliştirilme yapılması hedeflendiği için sadece iki amaçlı çalışmalar yapılmış ve karşılaştırmalar sadece BOA* ile yapılmıştır.

4.2.5 Diğer çalışmaların özetlenmesi:

Yukarıda sözü edilen çalışmaların kısa bir özeti Tablo-11'de yer almaktadır. Skriver'ın (1999) çalışması ise günümüzle karşılaştırınca çok zayıf kapasiteli bir bilgisayar üzerinde yapıldığı ve kullandığı haritalar diğer çalışmalarda kullanılan haritalara göre çok küçük kaldığı için bu tabloya alınmamıştır. Tablo-11 incelendiğinde, çalışmamızda ulaşılan sonuca benzer sonuçlara ulaşıldığı görülmektedir. Buna göre, BOA* yöntemi ve bu yöntemi baz alan yöntemler, iki amaçlı en kısa yol problemlerinde en hızlı çözümleri üretmektedirler. İki amaçlı en kısa yol probleminde BOA*'a göre yavaş kalan NAMOA* yöntemi ise ikiden çok amaçlı en kısa yol problemlerinde en iyi sonuçları üretmektedir. Keza düğüm ve kenar sayıları arttıkça sürelerin arttığı da görülmektedir.

Bütün çalışmaların sonuçlarını bir arada ortaya koyan bir tablonun oluşması önem arz etmektedir. Ancak sonuçların dikkatle incelenmesi gerekmektedir. Zira, her yıl kullanılan bilgisayarların kapasiteleri ve hızları artmaktadır. Bu ise çalışmamız gibi karşılaştırmalı çalışmaların önemini artırmaktadır, çünkü farklı yöntemler, aynı haritalar üzerinde, aynı bilgisayarlar kullanılarak karşılaştırılmaktadır. Öte yandan çalışmamızda yer alan süreler ile özellikle 2020 ve 2021'de yapılan çalışmaların süreleri karşılaştırıldığında, çalışmamızdaki sürelerin oldukça yavaş çıktığı görülmektedir. Bunun iki nedeni bulunmaktadır. Birincisi kullanılan bilgisayarlara ilişkindir. Sözü geçen çalışmalarda kullanılan bilgisayarlar Intel XEON işlemcili ve 128 GB RAM'e sahip serverlardır. Çalışmamızda kullanılan bilgisayarlar ise Intel i7 işlemcili ve 16 GB RAM'e sahip PC'lerdir. İkinci nedeni ise kullanılan programlama dilidir. Sözü geçen çalışmalarda C programlama dili kullanılmıştır. Çalışmamızda ise, gelecekte yapılması planlan geliştirilmeler göz önüne alınarak da, Python programlama dili kullanılmıştır. Bu iki nedenle, çalışmamızda yer alan süreler, diğer çalışmalara göre daha uzundur.

Tablo 11. Diğer çalışmalar karşılaştırma özet tablosu

Harita	Dimacs NY	Dimacs Bay Area	Dimacs Colorado	Dimacs Florida	Dimacs NE
Ulloa et al (2020) BOA*	0.32 sn	0.29 sn	0.79 sn	4.59 sn	-
Ulloa et al (2020) NAMOA*	152.17 sn	58.87 sn	476.26 sn	812.48 sn	-
Pulido et al. (2015) NAMOA*	21.957 sn	-	-	-	-
Ahmadi et al (2021) BOA*	0.12 sn	0.19sn	0.54 sn	2.05 sn	4.79sn
Ahmadi et al (2021) BOBA*	0.08 sn	0.13 sn	0.34 sn	1.31 sn	3.41 sn
Goldin ve Salzman (2021) BOA*	0.17 sn	-	-	-	25.8 sn
Goldin ve Salzman (2021) P-PA*	0.07 sn	-	-	-	1.3s

Yukarıda sözü edilen çalışmalar dikkate alındığında karşılaştırmamızda kullanılan NAMOA*, BOA* ve etiket düzeltme yöntemlerinin seçiminin doğru olduğu ortaya çıkmaktadır. Zira 2000'li yılların başına kadar geliştirilen kesin yöntemler içinde en hızlısının etiket düzeltme olduğu belirlenmiştir. Çok amaçlı yöntemler içinde NAMOA*, literatürde en çok karşılaştırma amacıyla kullanılan yöntemdir. Bu nedenle bu yöntem de karşılaştırmaya alınmıştır. Son olarak BOA* ve bu yöntemi esas alan yöntemler, bildiğimiz kadarıyla, çift amaçlı en kısa yol problemini en hızlı çözen yöntemdir. Bu nedenle de bu yöntem diğer yöntemler ile karşılaştırmaya alınmıştır.

DOI: 10.56554/jtom.1075286

5. Sonuç

Bu çalışmamızda, iki amaçlı en kısa yol problemi ve literatürde bu problemin çözümü için geliştirilmiş olan yöntemler incelenmiş ve karşılaştırılmıştır. Literatürde 1980'lerden beri geliştirilen birçok yöntem mevcuttur. Bu yöntemler kesin yöntemler ve sezgiseller olarak ikiye ayrılmaktadır. Çalışmamızda bu yöntemlerin en çok kullanılanları açıklanmıştır. Bu bağlamda problemi çalışacak olan araştırmacılar için bir başvuru kaynağı olmaktadır. Bildiğimiz kadarıyla da hem sezgiselleri hem de kesin yöntemleri bu ayrıntıda anlatan başka bir çalışma bulunmamaktadır.

Yukarıda da bahsettiğimiz üzere, iki amaçlı en kısa yol probleminin çözümünde birçok yöntem kullanılmıştır. Çalışmamızda açıklanan bu yöntemlerden üç tanesi çeşitli boyutlarda haritalar kullanılarak uygulama yapılmıştır. Seçilen yöntemlerden ilki kesin yöntemlerin en hızlısı olan etiket düzeltme yöntemidir. Bu yöntem ile Pareto optimal çözümlerin tamamı elde edilebilmektedir. Ancak bu yöntem harita boyutları büyüdükçe yavaş kalmaktadır. Kullandığımız ikinci yöntem 2010 yılında geliştirilen NAMOA* yöntemidir. Bu yöntem A* tabanlı bir sezgiseldir. Bu yöntem iki veya daha fazla amaçta çalışmaktadır. Bu nedenle sadece çift amaca mahsus değildir. Etiket düzeltmeye göre daha hızlı çalışan bu yöntem, 2020 yılında geliştirilen ve sadece iki amaçta kullanılan BOA* yöntemine göre oldukça yavaş kalmaktadır. BOA* yöntemi ise şu anda sadece iki amaca uygun olarak geliştirilmiştir. Çalışmamızda bu yöntemler, literatürde sıklıkla kullanılan veri setleri ile test edilmiştir. Bu anlamda çalışmamız, kesin yöntemler ile sezgisel yöntemleri karşılaştıran bir çalışma olmuştur. Çalışmamızda yer verdiğimiz diğer çalışmalarda ise daha çok kesin yöntemleri kendi içinde, sezgisel yöntemleri kendi içlerinde değerlendirilmişlerdir.

Sonuç olarak karşılaştırmasını yaptığımız yöntemlerden NAMOA* algoritmasının ikiden çok amaçlı problemlerde, BOA* algoritmasının ise iki amaçlı problemlerde kullanılmasının doğru olduğu değerlendirilmektedir.

Çalışmamızda, yöntemlerin çözüm sürelerinin kenar ve düğüm sayıları arttıkça arttığı fakat sadece bu amaçlara bağlı olmadığı görülmüştür. Çalışmamızda yapılmış olan uygulamada, kenar/düğüm oranı büyüdükçe, çözüm süresinin de değiştiği görülmüştür. Ancak buna ilişkin kesin bir kanıya varabilmek için başka uygulamalara da ihtiyaç duyulmakta ve ilerleyen çalışmalarda daha farklı amaçların de bulunabileceği düşünülmektedir. Gelecekteki çalışmalara ilişkin ortaya çıkan başka bir öneri ise çözüm yöntemlerinin en küçük toplam-en küçük azami değer veya en küçük azami değer-en küçük azami değer tarzı problemlere uyumlaştırılmasıdır.

Araştırmacıların Katkısı

Bu araştırmada Hakan Gürsoy, literatür taraması yapılması, kaynaklar üzerinden ikincil verilerin toplanması ve uygulama için hazırlanması, program kodlarının yazılıp çalıştırılması, verilerin özetlenmesi ve makalenin taslağının yazılması, Ekrem Duman, bilimsel yayın araştırmasının yeterliliğinin incelenmesi, yöntem ve uygulamanın incelenmesi, detaylı makale incelemesi ve yazıma ilişkin yönlendirme konularında katkı sağlamışlardır

Çıkar Çatışması

Bu makalenin yazarları arasında herhangi bir çıkar çatışması beyan edilmemiştir. Çalışmada yayın etiğine ilişkin kurallara uyulmuştur.

DOI: 10.56554/jtom.1075286

Kaynaklar

Akçayol, M. A. , S. Toklu. "Genetic algorithm based a new algorithm for time dynamic shortest path problem." *Journal of the Faculty of Engineering and Architecture of Gazi University*, December 2011. (https://dergipark.org.tr/tr/download/article-file/76015)

Ahmadi, Saman , Tack Guido ; Harabor, Daniel ; Kilby, Philip, "Bi-Objective Search with Bi-Directional A*", *29th Annual European Symposium on Algorithms,* 2021. (https://drops.dagstuhl.de/opus/volltexte/2021/14584/pdf/LIPIcs-ESA-2021-3.pdf)

Ahuja, R.K. T. L., Magnanti , J. B. Orlin. Network Flows: MIT Working Paper No: 2059-88, Prentice Hall, 1988. Doi : https://doi.org/10.1137/1037020

Arslan, H. , M. Manguoğlu. "A hybrid single-source shortest path algorithm", *Turkish Journal of Electric Engineering and Computer Sciences*, 2019 doi:10.3906/elk-1901-23

Bellman, E. "On a Routing Problem", *Appl. Math*., Vol 16, 87–90, 1958. (https://www.jstor.org/stable/43634538)

Brumbaugh-Smith, J Shier D. "An empirical investigation of some bicriterion-shortest path algorithms". *European Journal of Operational Research,* 1989, https://doi.org/10.1016/0377-2217(89)90215-4

Cheikh, Mohamed , Jarboui Bassem , Loukil Taicir. "A genetic algorithms to solve the bicriteria shortest path problem", *Electronic Notes in Discrete Mathematics,* 2010., https://doi.org/10.1016/j.endm.2010.05.108

Climaco, J.C.N. , E.Q.V. Martins. "A bicriterion shortest path algorithm" *European Journal of Operational Research*, 1982, https://doi.org/10.1016/0377-2217(82)90205-3

Dantzig, G.B., "On the Shortest Route Through a Network", *Management Science*, 187–190, 1960.

Dijkstra, E.W., "A Note on Two Problems in Connection with Graphs", *Numerische Mathematik,* Vol 1, 269–271, 1959, https://doi.org/10.1007/BF01386390

Ehrgott, M. Multicriteria Optimization 2nd edition, Springer , 2005, https://link.springer.com/book/10.1007/3-540-27659-9

Erhan Erkut, Vedat Verter, "Modeling of Transport Risk for Hazardous Materials". *Operations Research* 46(5):625-642., 1998 https://doi.org/10.1287/opre.46.5.625

Floyd, R.W. "Algorithm 97: Shortest Path", *Communications of the ACM 5*, 1962, https://doi.org/10.1145/367766.368168

Ford, L.R. Jr. "Network Flow Theory, Paper P-923", *The RAND Corporation*, Santa Monica, California, 1956.

Hansen, P. "Bicriterion Path Problems", *Multiple Criteria Decision Making Theory.* e Springer- Verlag, Berlin 1980., DOI: 10.1007/978-3-642-48782-8_9

Hart, P. E., N.J. Nilsson, B. Raphael. "A Formal Basis for the Heuristic Determination of Minimum Cost, Paths". *IEEE Transactions on Systems Science and Cybernetics*, 1968. https://doi.org/10.1109/TSSC.1968.300136

Mandow, L. , J.L. Pérez de la Cruz "Multiobjective A* search with consistent heuristics." *Journal of the ACM 57 (5)*, 2010, https://doi.org/10.1145/1754399.1754400

Mandow L., Pulido, F.J. , J. L. Perez-de-la-Cruz. "Dimensionality reduction in multiobjective shortest path search" *Computers , Operations Research 64:60–70.* , 2015, https://doi.org/10.1016/j.cor.2015.05.007

Mandow L., Pulido, F.J. , J.L.Perez-de-la-Cruz."Multiobjective shortest path problems with lexicographic goal-based preferences". *European Journal of Operational Research 239*, 2014, https://doi.org/10.1016/j.ejor.2014.05.008

DOI: 10.56554/jtom.1075286

Martins, E.Q.V., On a "Multicriteria Shortest Path Problem", *European Journal of Operational Research*,1984., https://doi.org/10.1016/0377-2217(84)90077-8

Murthy, J. I. Mote , D.L. Olson. "A parametric approach to solving bicriterion shortest path problems", *European Journal of Operations Research* 53, 1991,

Raith, Andrea, Matthias Ehrgott. "A comparison of solution strategies for biobjective shortest path problems", *Computers , Operations Research 36,* 2009, https://doi.org/10.1016/j.cor.2008.02.002

Schrijver A. "On the History of the Shortest Path Problem", *Documenta Mathematica*, 2012.

Sedgewick, Robert , K. Wayne. *Algorithms, Addison Wesley Publishing,* 2014.

Serafini, P. (1986). "Some considerations about computational complexity for multi objective combinatorial problems." *Recent advances and historical development of vector optimization, volume Bibliograph of Lecture Notes in Economics and Mathematical Systems,* Springer Verlag, Berlin.

Shi, Ning , Shaorui Zhou , Fan Wang , Yi Tao , Liming Liu. "The multi-criteria constrained shortest path problem", *Transportation Research Part E*, 2017, https://doi.org/10.1016/j.tre.2017.02.002

Skriver, A. "A Classification of Bicriterion Shortest Path Algorithms", *Asia Pasific Journal of Operational Research* , 2000a.

Skriver, A. , K. A. Andersen, "A Label Correction approach for solving bicriterion shortest-path problems", *Computers , Operations Research*, 2000b, https://doi.org/10.1016/S0305-0548(99)00037-4

Stewart, B.S. , C.C. White. "Multiobjective A*". *Journal of the ACM 38 (4)*, 1991, https://doi.org/10.1145/115234.115368

Tung, CT , K.L. KL. "A bicriterion Pareto-optimal path algorithm". *Asia-Pacific Journal of Operations Research,* 1988, https://doi.org/10.1016/0377-2217%2892%2990248-8

Ulloa, C.H. , W. Yeohz , J. Baier , H. Zhang , L. Suazo , S. Koenig. "A Simple and Fast Bi-Objective Search Algorithm", *Association for the Advancement of Artificial Intelligence,* 2020, https://doi.org/10.1609/icaps.v30i1.6655

Xiao-Bing, Hu , Chi Zhang Gong , Peng Zhang Ming , Kong Zhang , Hang Li , Mark S. Leeson, Jian-Qin Liao. "Finding the *k* shortest paths by ripple-spreading algorithms Finding the *k* shortest paths by ripple-spreading algorithms", *Artificial Intelligence,* 2020, https://doi.org/10.1016/j.engappai.2019.08.023

Yen, Jin Y., "Finding the K Shortest Loopless Paths in a Network", *Management Science*, 1971

https://en.wikipedia.org/wiki/Yen%27s_algorithm (erişim tarihi: 18.06.2021)

Shortest Path Data, Princeton University, " https://algs4.cs.princeton.edu/44sp/" (Erişim tarihi: 18.06.2021)

Dimacs (2005), Center for Discrete Mathematics and Theoretical Computer Science, "http://www.diag.uniroma1.it//~challenge9/download.shtml" (Erişim tarihi: 18.06.2021)

DOI: 10.56554/jtom.1075286