JLECON

Volume / Cilt: 10, Issue / Sayı: 4, 2023, pp. 247-256
E - ISSN: 2148-4139
URL: https://www.journals.gen.tr/jlecon
DOI: https://doi.org/10.15637/jlecon.2219

Received / Geliş: 30.08.2023
Acccepted / Kabul: 02.10.2023

RESEARCH ARTICLE / ARAŞTIRMA MAKALESİ

Aylak zamanı enküçükleyen tur oluşturma problemlerinin geri izleme yöntemi ile çözümüne ilişkin bir yazılım geliştirme uygulaması

A software development application for solving round trip slack time minimizing problems using backtracking method

Şahin İnanç[1] **Hayrettin Kemal Sezen**[2]

[1] Öğr. Gör. Dr., Uludağ Üniversitesi, Keles Meslek Yüksekokulu, Bilgisayar Teknolojileri Bölümü, Bilgisayar Programcılığı Programı, Türkiye, e-mail: sahininanc@uludag.edu.tr

[2] Prof. Dr., Altınbaş Üniversitesi, Uygulamalı Bilimler Fakültesi, Yönetim Bilişim Sistemleri Bölümü, Türkiye, e-mail: kemal.sezen@altinbas.edu.tr

Öz

İşletmelerde gün geçtikçe artan rekabet koşullarında küçük avantajlar bile önemli olabiliyorken lojistik de her geçen gün daha da büyük bir öneme sahip olmaktadır. İşletmelerin giderlerinin önemli bir kısmını lojistik oluşturmaktadır. Bu çalışma karayolunda yolcu taşıması yapan bir lojstik şirketinin araçlarının seferleri arasında beklemelerine ilişkin aylak zaman toplamını en küçük kılacak şekilde, optimal çözümü garanti eden bir (kesin, exact) yöntemle çözümü gerçekleştirilmiştir. Çözüme yönelik C# programlama dili ile bir yazılım geliştirilmiştir. Problemin çözümüne ilişkin yöntem olarak Dal ve Sınır tekniğinin Geri İzleme yaklaşımı kullanılmıştır. Kullanılan bu yöntem; tam sayıma göre sayımlamayı azaltma özelliğine sahiptir.

Anahtar Sözcükler: Araç Rotalama Problemi, Gezgin Satıcı Problemi, Dal ve Sınır Yöntemi, Geri İzleme Yöntemi, C#

JEL: kodları C0, C60, C61.

Abstract

Even a small advantages can be important in the competition conditions that are increasing day by day in the enterprises, logistics is becoming more important with each passing day. A significant part of the expenses of enterprises constitute logistics. This study was carried out with a (exact, exact) method, which guarantees the optimal solution so as to minimize the total time spent waiting for the vehicles of a lojstik company carrying road passenger. The software was developed with the C# programming language for the solution. The method of solution of the problem was followed by the Backtracking approach of the Branch and Bound technique. This method used; has the ability to reduce total completed counting.

Keywords: Vehicle Routing Problem, Traveling Salesman Problem, Branch and Bound Technique, Backtracking Approach, C#

JEL codes: C0, C60, C61.

Citation/Atıf: İNANÇ, Ş. & SEZEN, H. K. (2023). Aylak zamanı enküçükleyen tur oluşturma problemlerinin geri izleme yöntemi ile çözümüne ilişkin bir yazılım geliştirme uygulaması. *Journal of Life Economics*. 10(4): 247-256, DOI: 10.15637/jlecon.2219

Corresponding Author/ Sorumlu Yazar:
Şahin İnanç
E-mail: sahininanc@uludag.edu.tr

1. GİRİŞ

Gün geçtikçe değişen ve gelişen küresel rekabet ortamında işletmelerin varlıklarını sürdürmesi ve rekabet avantajı elde etmesi için maliyet azaltma ile ilgili çözüm yolları aramasıyla, ulaştırma konusu giderek önem kazanmış, önemli bir rekabet unsuru haline gelmiştir. Sınırlı mallar ve ulaşım kaynakları, yüksek planlama karmaşıklığı ve lojistik hizmet sağlayıcıları arasındaki güçlü rekabet sayesinde artan maliyet baskısı, (Caric vd, 2008: 2) müşteri hizmetini optimize etmek için entegre lojistik sistemleri birincil ihtiyaç haline gelmiştir. İşletmeler ulaştırma ve taşıma maliyetlerini minimize etmek, aynı zamanda müşteri ihtiyaçlarının zamanında ve tam olarak karşılanması gibi birbiri ile çelişen problemlerin çözüm yolu arayışına girmiştir. Bu gibi karmaşık problemlerin çözümünde en çok kullanılan yöntemlerden biri, bir dağıtım şebekesinde araç filosuyla bir dizi müşteriye hizmet etmek isteyen bir kombinasyonel optimizasyon problemi olan "Araç Rotalama Problemi" optimizasyonudur. Araç Rotalama Problemi (ARP), bir veya birkaç depodan, coğrafi olarak dağınık şehirlere veya müşterilere, yan kısıtlamalara tabi olmak üzere, en uygun teslimat veya toplama güzergâhlarının tasarlanması problemi olarak tanımlanabilir. (Laporte, 1992: 1). Araç Rotalama Problemlerinde (ARP) amaç genellikle zamanı ya da mesafeyi veya birden çok kısıtı birden minimize etmektir. Bu kısıtlara bağlı kalınarak müşterilerin ihtiyaçları vaktinde karşılanmaya çalışılır. Bu sorun, işletmelerin bir dizi coğrafi olarak dağılmış müşteriye ulaştırılması için bir teslimat filosunun sağlanmasıyla ilgili zaman ve maliyetler nedeniyle işletmeler için ekonomik açıdan önemlidir. Buna ek olarak, bu tür sorunlar, otobüs sistemlerinde, posta taşıyıcılarında ve diğer kamu hizmet araçlarında araç rotalarının belirlenmesini gerektiren kamu sektöründe de önemlidir. Bu örneklerin her birinde, sorun tipik olarak bir tedarik yerinden bir takım müşteri lokasyonlarına teslimatı kolaylaştırmak için bir takım araçlar için birleştirilmiş yolların minimum maliyetini bulmayı içerir. Maliyet, mesafe ile yakından ilişkili olduğundan, bir şirket müşteri talebini karşılamak için birtakım araçların kat ettiği asgari mesafeyi bulmaya çalışabilir. Bunu yaparken, firma, beklenen müşteri hizmet düzeyini artırırken veya en azından koruyarak maliyetleri en aza indirmeye çalışır (Bell vd, 2004: 41).

ARP ilk olarak George Bernard Dantzig ve John Ramser tarafından 1959 yılında ele alınmıştır. Dantzig ve Ramser çalışmalarında ayrı yerlerde bulunan servis istasyonlarına akaryakıt dağıtım problemini ele alıp çözüm için bir matematiksel programlama modeli geliştirmişler ve algoritmik bir yaklaşım ortaya koymuşlardır (Dantzig vd, 1959).

1.1. Literatüre Kısa Bakış

Hokama vd. (2016), çalışmalarında, araç rotalama problemlerini boşaltma kısıtı altında Dal Kesme yöntemini kullanarak problemin çözümü için algoritma geliştirmişlerdir.

Montoya vd. (2015), çalışmalarında, yeşil araç rotalama problemi (Yeşil ARP), alternatif yakıt araçları (AYA'ler) kullanılarak rotalama problemini çözmüşlerdir. Problemde AYA'ler sınırlı tank kapasitesine sahip olduğundan, güzergâhlar sadece alternatif yakıt istasyonlarından (AYİ'ler) geçebilirler. Montoya vd. yeşil ARP ile başa çıkmak için basit ama etkili iki aşamalı bir buluşsal yöntem önermişlerdir.

Barkaoui vd. (2015), çalışmalarında, müşteri memnuniyetini artırmaya yönelik bir strateji sunmuşlardır. Önerilen yeni yöntemde, daha önce ziyaret edilmiş müşterilerin memnuniyetini artırmak için zaman penceresiyle dinamik araç rotalamanın birleştirilmesi ile tasarlanmış hibrit bir genetik algoritma kullanılarak gerçekleştirilmiştir. Simülasyonlar, yeni stratejiyi kullanan revize edilmiş algoritmanın değerini karşılaştırmakta ve bunun müşteri memnuniyeti üzerindeki etkisini açıkça göstermektedir.

Dastghaibifard vd. (2008), çalışmalarında, ARP için yeni bir paralel Dal ve Sınır algoritması önerilmiştir. Bu algoritmada, çok işlemcili paylaşılan bellek yerine çok bilgisayarlı ve dinamik yük dengeleyici bir yaklaşım kullanılmış. Problem türü olarak kapasiteli ARP seçilip yeni yöntem denenmiştir. Buldukları sonuçlar diğer algoritmalara göre daha başarılı olduğu görülmektedir.

Liu vd. (2008), çalışmalarında, basit montaj hattı dengeleme tip I probleminin çözümü için kesin

algoritmalar önermişlerdir. Önerilen algoritmalar yapıcı ve iki yıkıcı algoritmadan oluşuyormuş. Bu algoritmalarda, iyi bilinen birkaç alt sınır hesaplama yöntemi de uygulanmış. Bir dizi kıyaslama problemi örneğine dayanarak, önerilen algoritmaların performansını test etmek için hesaplamalı deneyler yapılmış. Hesaplanan sonuçlar, geliştirilen algoritmaların basit montaj hattı dengeleme kıyaslama problemi örneklerini çözmede etkili olduğu görülmektedir.

2. ARAÇ ROTALAMA PROBLEMLERİ

2.1. Dal ve Sınır Yöntemi

İlk olarak 1950'li yıllarda A.H.Land ve A.G. Doing tarafından geliştirilen Dal ve Sınır Yöntemi (D & S) matematiksel programlama problemlerinin en iyi çözümlerinin bulunması ile ilgili pek çok yöneylem araştırması probleminde uygulanmaktadır. Doğrusal Programlama problemlerinin tamsayı karar değişkenleri ile çözülmesi bağlamında Land ve Doig (1960) Dal ve Sınır metodunu geliştirmiştir. Tamsayılı Programlama modelini standart Doğrusal Programlama teknikleri (örneğin, simpleks metodu) kullanarak çözmeye çalışırken, tamsayı değerlerini almak için gerekli olan değişkenlerden bir veya daha fazlası, gerçek Doğrusal Programlama çözümünde kesirli olabilir. Dallanma Land ve Doig'ın algoritma dallarındaki süreçleri bu kesirli değişkenlerden alır. Bir dal, kesirli değişkenin, en büyük tamsayıya eşit veya daha küçük bir değere sahip olmasını gerektirir, diğer dal ise, değişkenin, daha büyük olan en küçük tam sayıdan büyük veya ona eşit bir değer almasını gerektirir. Algoritma, dalların düğümleri için doğrusal programlama alt problemlerinin çözümü ile devam eder. Nihai amaç, Doğrusal Programlama problemine en uygun çözümü üretmektir (Brusco, 2005: 4).

Dal ve Sınır yöntemi bir sorunla ilgili türlü aşamaları sistemli bir şekilde analiz ederek en iyi çözümü araştırır. Dal ve Sınır (D & S) algoritmaları, operasyonel araştırma, kombinasyonel optimizasyon ve yapay zeka konularında çok çeşitli problemleri çözmek için yaygın olarak kullanılır. Ancak, hangi algoritmaların D & S kategorisine girdiğine dair birçok fikir ayrılığı vardır. "Dal ve Sınır" terimi, genel, Karma-Tamsayı, Doğrusal Programlama problemlerini çözmek için bir yönteme başvurmak için ilk olarak operasyonel araştırma alanında kullanılmıştır. Tartışmalı olarak, son yıllarda, yapay zekâ alanında kullanılan bir dizi AND / OR grafik arama prosedürünün aslında D & S algoritmaları olarak görülebileceği iddia edilmiştir (McKeown, 1991: 1).

Başlangıç adımda olanaklı çözümlerin toplam kümesi daha küçük alt kümelere ayrılır. Aynı anda her bir alt küme için üst veya alt sınır değerleri belirlenir. Daha sonra bu değerlere bağlı olarak bazı alt kümelerin çözümden atılması işlemi gerçekleştirilir (Sezen, 2017: 120).

D & S'de arama alanı, kökü asıl sorun olan bir ağaç olarak temsil edilir, iç düğümler kısmen alt problemleri çözer ve yapraklar potansiyel çözümlerdir. D & S, şimdiye kadar bulunan en iyi çözümün (üst sınır) giderek geliştirildiği birkaç yinelemeyle ilerler: Daha iyi bir çözüme götürecek ve karşılık gelen alt dalları kesecek olan alt problemleri ortadan kaldırmak için bir sınırlama mekanizması kullanılır. Bu, keşfedilen arama alanının boyutunu azaltır, ancak pratikte hala zaman alıcı olabilir ve örneğin paralel hesaplama kullanarak hızlanma gerektirir (Borisenko, 2017: 640). Dal ve Sınır algoritmasında, Gezgin Satıcı Problemi, alt tur (depoda başlayıp bitmeyen turlar) engelleyici kısıtları yok edilerek atama problemi haline dönüştürülür ve Macar Yöntemi ile çözüme başlanır. Satır ve sütün eleme yöntemiyle rotalar belirlenmeye çalışılır. Alt tur oluşursa; en kısa döngüyü engelleyecek kısıtlar ile dallandırılır. İstenilmeyen rotalara atama yapılmaması için maliyet matrisinde ceza katsayısı olarak büyük M sayısı atandıktan sonra matris tekrar baştan çözülerek tüm dallar için aday çözümler belirlenir. Tüm dallar için aynı iterasyonlar tekrarlandıktan sonra en iyi çözüme karar verilir (Keskintürk, 2015: 88).

Dallanma rutini, eğer bir alt uzay ümit vaat ediyorsa ve atılamazsa uygulanır, dolayısıyla bu alt alan, sonraki iterasyonlarda keşfedilecek diğer alt alanlara bölünür (Kadri, 2016: 44).

Dallanma var olan problemi iki ayrı alt probleme ayırma işlemidir. Böylece asıl problem yerine iki ayrı alt problem ortaya çıkmış olur. Oluşturulan bu alt problemler asıl problemin parçalanmış alt problemleri olmaktadır. Bu parçaları asıl

problemin kısmen çözülmüş halleri şeklinde de düşünebiliriz. Problemi Dal ve Sınır yöntemi ile çözebilmek için her aşamada problem alt problemlere (dallara) ayrılır. Ayrılan bu dallar diğer aşamada tekrar alt problemlere (dallara) ayrılır. Dal ve Sınır yöntemi ile problemin çözümü için dallanma iki farklı şekilde olmaktadır. Bunlar permütasyonel dallanma ve kombinasyonel dallanma olarak isimlendirilir (Sezen, 2017: 121).

Sınırlama fonksiyonu, D & S algoritmasının verimliliğinin ana bileşenidir ve düğüm seçimi ve dallanması için uygulanan stratejilerle dengelenemez. En iyi durumda, belirli bir alt problem için bir sınırlama fonksiyonunun değeri, optimal çözümün değerine eşit olmalıdır, ancak bu amaca ulaşmak çoğu durumda zordur (Kadri, 2016: 44).

Dallarda sayımlamayı azaltmak ve optimal çözümü daha az sayımlama ile bulmak için sınırlama yapılır. Sınırlama yaparken sınırlama işlemi yapılacak alt problemin en iyi çözümü için önceden bir değer ya da aralık belirlenir. Bu aralığa ya da değere göre sınırlama yapılır. Yaratılan her bir alt problemde sınırlama yaparken genellikle alt problemin alabileceği en küçük ve en büyük değerler belirlenir. Belirlenen bu değerlerden en küçük olanı alt sınır, en büyük olanı da üst sınır olarak isimlendirilir. Alt ve üst sınırlar genellikle hesaplanarak bulunur. Sınırlama ile problemin alt çözümlerinden bazıları çözüm dışında bırakılarak sayımlama azaltılır. Bunun sonucunda bu probleme ait sınırlanmış alt dallar için gereksiz hesaplamalardan kaçınılmış olur. Başka bir deyişle sınırlama ile bir alt dalın dallandırılmasının verimli olup olmayacağı ortaya çıkmış olur. Bir alt dala konulan sınır ile problem çözümünün devamında bu dal üzerinden en iyi çözüme gidilip gidilemeyeceğine karar verilebilir (Sezen, 2017: 121).

Dal ve Sınır yönteminde optimal çözüm bulmak için iki farlı yol vardır. Bunlardan ilki Sıçramalı İzleme Yaklaşımı, diğeri de Geri İzleme Yaklaşımıdır.

2.1.1. Geri İzleme Yaklaşımı

Geri İzleme (Backtracking) Yaklaşımının temelinde yatan düşünce; ilk önce hızlı bir şekilde deneme çözümü bulunup daha sonra geriye doğru daha iyi çözüm olup olmadığının araştırılmasıdır. Diğer bir deyişle ağaç yapısı üzerinde bir deneme çözümü bulunup daha sonra bu çözümün adım adım iyileştirilmesidir. Bir problemde sayımlamayı azaltmanın yollarından biri deneme çözümü kullanmaktan geçer. Bir deneme çözümü bulunduktan sonra, en iyi çözüm olarak; bulunan deneme çözümü kabul edilir. Deneme çözümünün bulunmasında herhangi bir yöntem kullanılabilir. Deneme çözümü bulunmazsa Dal ve Sınır yöntemi ile çok fazla sayımlama yapmak gerekir. Sayımlamayı şu şekilde azaltabiliriz: Deneme çözümü bulduktan sonra alternatif çözümler için alt dallar araştırılırken daha kötü bir alt dala rastlandığında bu dalın sonraki aşamalarına (alt dallar) bakılmaksızın yarıda kesilerek çözümden atılır. Böylece daha az alt dal araştırılmış olup sayımlama azaltılır. Alt dallar araştırılırken ağacın en alt sınırına kadar gelindi ise ve bulunan çözüm deneme çözümünden daha iyi bir çözüm ise mevcut deneme çözümü yerine yeni bulunan çözüm geçer. Bu işlemler ağacın başlangıcından geriye doğru gidilerek tekrar edilir. Araştırılacak hiç alt dal kalmayıncaya kadar işlemlere devam edilir. Araştırılacak alt dal kalmayınca o ana kadar bulunan en son deneme çözümü en iyi çözüm olmuş olur (Sezen, 2017: 140).

Geri İzleme Yaklaşımı ile enküçükleme probleminin çözüm adımları şu şekildedir:

1. Adım: Asıl problem için alt sınır değerlerini belirle ve problemi alt problemlere (dallara) ayır.

2. Adım: Yeni yaratılan problemler için alt sınır değerlerini belirle.

3. Adım: En küçük sınır değerine sahip düğümü seçip alt problemlere parçalayıp dallandır.

4. Adım: Dallandırılacak düğüm kaldı ise 2.Adım'a git.

5. Adım: Dallandırma işleminin sonuna gelindiğinde bulunan çözüm değerini deneme çözümü olarak al.

6. Adım: Elde edilen deneme çözümü ile son aşamadan geriye doğru karşılaştırmalar yapılarak devam edilir. Yapılan karşılaştırmalarda sınır değeri daha büyük olan düğümler elimine edilir.

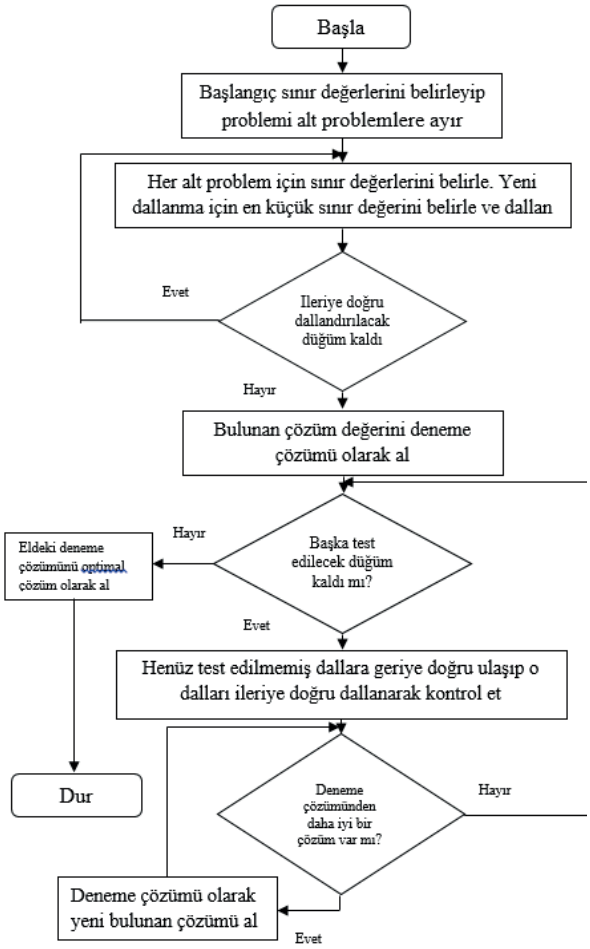
Daha sonraki aşamalarda bu düğümler üzerinde işlem yapılmaz.

7. Adım: Elimine edilemeyen düğüm değeri deneme çözümü değerinden küçük olduğu sürece en son aşamaya gelene kadar (tam bir çözüm bulununcaya kadar) dallandırmaya devam edilir. Yapılan karşılaştırmalarda daha küçük değerler bulundu ise deneme çözümü yerine daha küçük değerli yeni çözüm geçer.

8. Adım: Dallanma işlemi elimine edilmeyen tüm düğümlere uygulandıktan sonra optimal çözüm bulunmuş olur (Sezen, 2017: 141).

Geri İzleme Yaklaşımı ile enküçükleme probleminin çözüm adımları için akış diyagramı aşağıda yer alan şekil 1'deki gibidir:

Şekil 1. Geri izleme yaklaşımı akış diyagramı



3. UYGULAMA

3.1. Problemin Tanımı

Çalışmada yer alan problem araçların boşta kalma (aylak kalma) sürelerinin enküçüklemeye çalışıldığı, karayolunda yolcu taşıması yapan bir otobüs şirketine ilişkindir. Otobüs şirketinde kullanılan belirli bir sefer topluluğu için aylak zaman toplamını enküçükleyen rota belirlenmesine ilişkin optimal çözüm araştırılacaktır. Uygulamada rota dairesel olarak hesaplanmaya çalışılmıştır.

Diğer bir deyişle rota başladığı şehirde son bulacak şekilde hesaplanmaya çalışılmıştır. Çözümde alt turlar oluşabilmektedir. Bu durumda alternatif rotalar taranarak çözüme sadece dairesel olan rotalar dahil edilmiştir. Aynı şekilde çözümü imkânsız ya da çıkmaz rotalama durumlarıyla karşılaşılabilmektedir. Yine bu durumda da alternatiflere bakılıp çözümsüz rotalar çözümden çıkarılmış, dikkate alınmamıştır.

Uygulamada aynı şehre birden çok kez uğranabilmektedir. Problemde tanımlanan şekilde her şehre tanımlanan sefer sayısı kadar uğranılmıştır. Bu nedenle var olan problemdeki düğüm sayısı şehir sayısı kadar olmayıp sefer sayısı kadardır. Problemde bir şehre farklı zamanlarda tanımlanan her bir sefer için tekrar uğranacağından o şehir-zaman bileşimi ayrı bir düğüm olarak alınmalıdır.

3.2. Veriler

Uygulamada aşağıdaki iller ve seferler için aylak zamanı en küçük kılacak şekilde optimal çözüm araştırılmıştır. Tablo 1'de görüldüğü gibi 24 sefer için sefer numarası, seferin başlayacağı şehir, seferin sona ereceği şehir, kalkış zamanı ve yolculuk süresi gibi bilgiler eldeki problem için tanımlanmıştır.

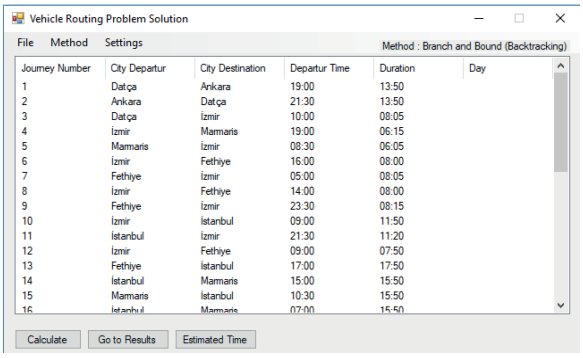
Tablo 1. Uygulama verileri

Sefer No.	Başlangıç Şehri	Bitiş Şehri	Kalkış Saati	Yol. Süresi
1	Datça	Ankara	19:00	13:50
2	Ankara	Datça	21:30	13:50
3	Datça	İzmir	10:00	08:05
4	İzmir	Marmaris	19:00	06:15
5	Marmaris	İzmir	08:30	06:05
6	İzmir	Fethiye	16:00	08:00
7	Fethiye	İzmir	05:00	08:05
8	İzmir	Fethiye	14:00	08:00
9	Fethiye	İzmir	23:30	08:15
10	İzmir	İstanbul	09:00	11:50
11	İstanbul	İzmir	21:30	11:20
12	İzmir	Fethiye	09:00	07:50
13	Fethiye	İstanbul	17:00	17:50
14	İstanbul	Marmaris	15:00	15:50
15	Marmaris	İstanbul	10:30	15:50
16	İstanbul	Marmaris	07:00	15:50
17	Marmaris	İstanbul	00:00	15:50
18	İstanbul	Fethiye	17:00	17:50
19	Fethiye	İzmir	12:30	08:05
20	İzmir	Alanya	23:00	10:50
21	Alanya	İzmir	10:00	12:20
22	İzmir	Fethiye	00:00	08:00
23	Fethiye	İzmir	08:30	08:05
24	İzmir	Datça	18:00	07:35

2.3. Yazılımın Ekran Görünümü

Geliştirilen yazılımda üç *Form* penceresi yer almaktadır. İlk pencerenin ekran görünümü şekil 2'deki gibidir.

Şekil 2. Yazılımın ekran g**örünümü**

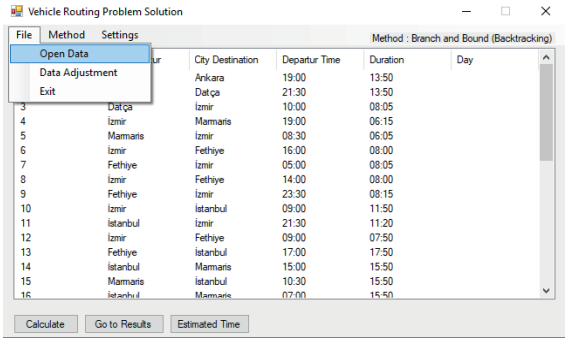


Şekil 2'de otobüs seferleri ile ilgili ayrıntılı bilgiler görünmektedir. Bu bilgilerin anlamları aşağıda yer alan tablo 2'de gösterilmiştir.

Tablo 2. Ekran görünümündeki bilgiler

Ekran Görünümündeki İsim	Anlamı
Journey Number	Sefer Numarası
City Departur	Kalkış Şehri
City Destination	Hedef Şehir
Departur Time	Kalkış Zamanı
Duration	Süre
Day	Gün

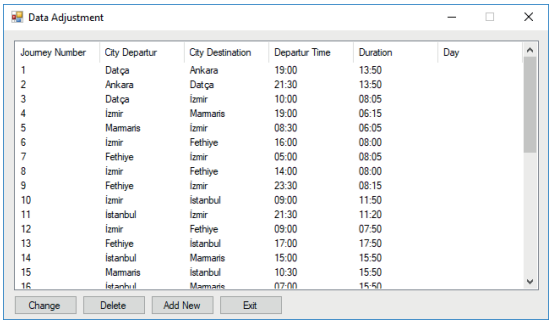
Şekil 3. Ekran görünümündeki file (dosya) menüsü



Buradaki veriler Access veri tabanından çekilmektedir. Farklı bir veri tabanından veri çekmek istediğimizde şekil 3'deki gibi *File* menüsünden *Open Data* seçeneğini seçiyoruz. Buradan açılan pencere ile veri tabanını seçip verileri ekrana yansıtıyoruz.

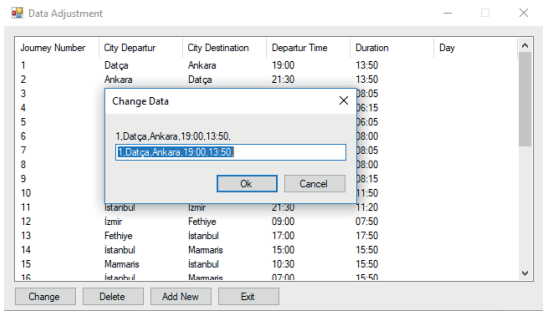
İkinci *Form* penceresi veriler üzerinde değişiklik yaptığımız penceredir. Bu pencereyi *File* menüsünden *Data Adjustment* seçeneğini seçerek açıyoruz. Bu seçenekten sonra şekil 4'teki gibi yeni bir pencere karşımıza çıkıyor. Bu pencereden sadece veriler üzerinde değişiklik yapılabiliyor. Bu değişiklikler var olan veriyi değiştirme, var olan veriyi silme, yeni bir veri ekleme şeklindedir. Değişiklikleri şekil 3'de görüldüğü gibi *Change* butonuna basarak değiştirebileceğimiz gibi pencere üzerinde yer alan verinin üzerine çift tıklayarak da yapabiliriz. Verinin üzerine çift tıkladığımızda açılan pencere şekil 4'de görünmektedir.

Şekil 4. Data adjustment ekran görünümü



Bunların dışında *Delete* butonuna bastığımızda o an seçili olan veri silinir.

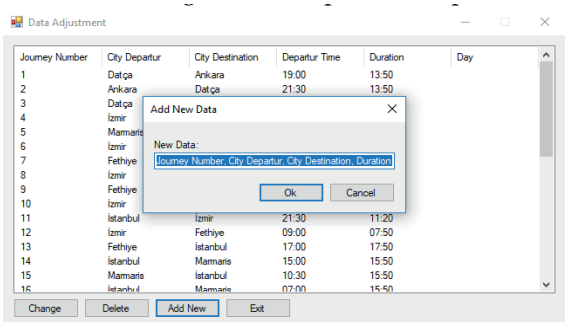
Şekil 5. Change data ekran görünümü



Add New butonuna bastığımızda ise altta yer alan şekil 6'daki gibi küçük bir pencere açılmaktadır. Bu pencereye veri eklerken veriler arasında virgül bulunması gerekir. Buraya yazdığımız veriden sonra *Ok* butonuna bastığımızda pencerede yer alan veri satırı verilerin var olduğu veri tabanına yeni bir kayıt olarak eklenecektir. *Cancel* butonuna bastığımızda ise veri ekleme işlemi iptal edilecektir.

Data Adjustment penceresinde yer alan *Exit* butonuna bastığımızda ise pencere kapanır.

Şekil 6. Add new data ekran görünümü

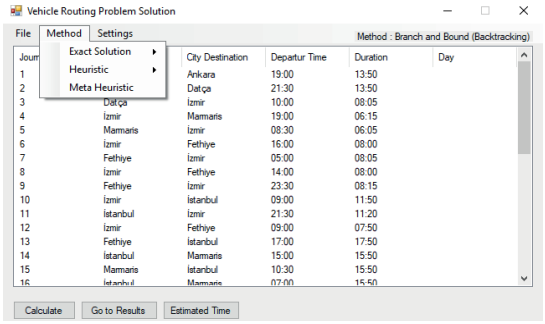


Herhangi bir verinin üzerine çift tıkladığımızda yukarıda yer alan şekil 6'daki gibi bir pencere karşımıza çıkmaktadır. Bu pencerede üzerine çift tıklanan verinin ayrıntıları görünmektedir. Burada veriler arasında virgül olacak şekilde istediğimiz kısımları silip değiştirdiğimizde ve ardından *Ok* butonuna bastığımızda değişiklikler veri tabanına kaydedilip *Data Adjustment* penceresine otomatik yansıtılacaktır. *Cancel* butonuna basıldığında ise pencerede değişiklik yapılsa da veri tabanına ekleme yapılmayıp herhangi bir değişiklik olmayacaktır.

File menüsünde *Exit* seçeneği ise programdan çıkılmasını sağlar.

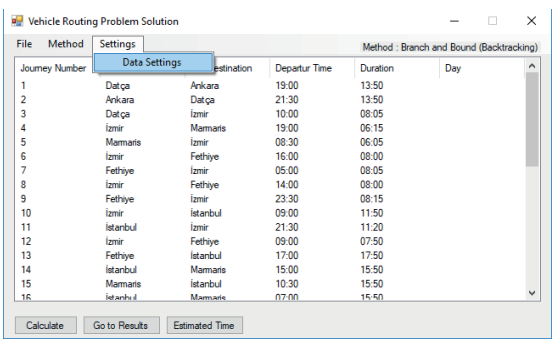
Aşağıda yer alan şekil 7'de görüldüğü gibi menü seçeneklerinden *Method* seçeneği seçildiğinde ise Araç Rotalama Problemlerinin (ARP) çözümü ile ilgili seçenekler görülmektedir. Buradan seçilen yönteme göre farklı çözüm yöntemleri ile ilgili seçenekler yer almaktadır. Buradan seçilen yöntem ilk pencerede sağ üstte görülmektedir. Örnek pencerede *Method: Branch and Bound (Backtracking)* (Yöntem: Dal ve Sınır (Geri İzleme)) olarak görülmektedir.

Şekil 7. Method sekmesi ekran görünümü



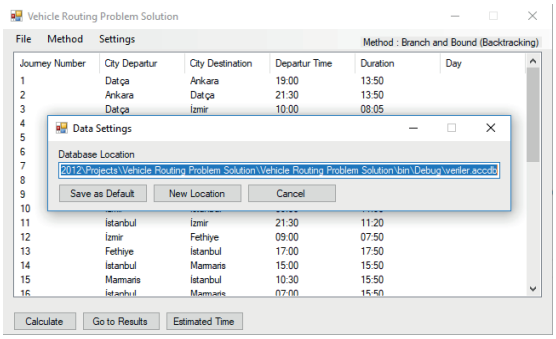
Menü seçeneklerinden *Settings* altında yer alan *Data Settings* ise veri tabanı için geçerli konum ayarlamasını yapmak için kullanılmaktadır. Program tekrar açıldığında, hangi konum ayarlandı ise veri tabanın o konumundan verileri çekip programı açmaya çalışacaktır. Bu seçenek ile ilgili ayrıntılar şekil 8'deki gibidir.

Şekil 8. Data settings sekmesi ekran görünümü



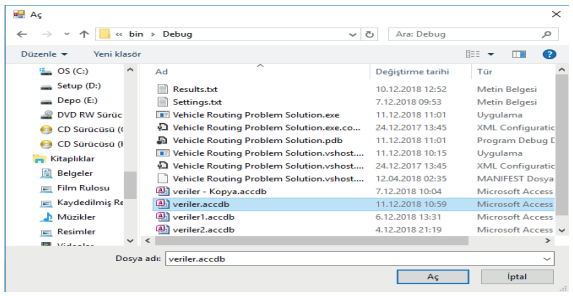
Data Settings'e tıklandığında şekil 9'deki pencere açılacaktır.

Şekil 9. Data settings ekran görünümü



Burada *Database Location* olarak görünen veri tabanının geçerli konumunu değiştirebiliriz. *Save as Default* butonuna bastığımızda üstteki *Database Location* olarak yer alan adres ne ise o adres geçerli adres olarak kaydedilecektir. *New Location* butonuna bastığımızda aşağıda yer alan şekil 10'deki gibi yeni bir konum adresi için bir pencere açılacaktır. Bu açılan pencereden hangi veri tabanı dosyasını seçersek *Aç* butonuna bastığımızda o veri tabanı konumu *Database Location* olarak görünecektir. İptal butonuna bastığımızda ise konum ile ilgili değişiklik iptal edilmiş olacaktır. Daha sonra yine *Save as Default* butonuna basarak yeni veri tabanı dosyası için konumu kaydetmemiz gerekir.

Şekil 10. Adres konumu ekran görüntüsü

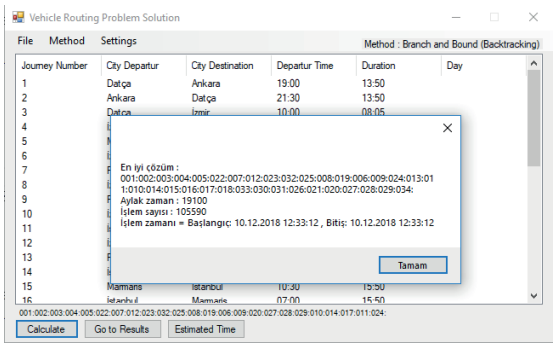


Cancel butonuna bastığımızda ise veri tabanı konumu ile ilgili değişiklik iptal edilmiş olacaktır. Veri tabanı konumu ile ilgili bilgiler programın bulunduğu konumda *Settings.txt* adlı metin belgesine kaydedilmektedir. Bu metin belgesinden konum ile ilgili bilgiyi metin belgesini açarak elle de değiştirebiliriz.

İlk pencerede şekil 11'deki gibi yer alan *Calculate* butonuna bastığımızda ise sağ üstte yer alan *Method: Branch and Bound (Backtracking)* olarak ne seçildi ise o yönteme göre program problemin çözümünü bulmaya çalışacaktır.

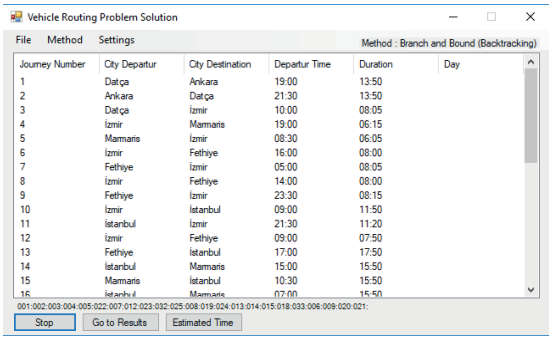
Şekil 11'de *Calculate* butonuna basıldığında değişiklikler görülmektedir. *Calculate* butonunun ismi *Stop* olarak değişmektedir. *Stop* butonuna basıldığında ise hesaplama yarıda kesilecek ve *Stop* butonu ismi tekrar *Calculate* olacaktır. Butonun hemen üstünde yer alan

Şekil 11. Calculate butonu ekran görünümü



001:002:003: … gibi bilgiler olası rotaları göstermektedir. Bu rotalardan büyük bir kısmı dairesel rota oluşturmamaktadır. Bir kısmı imkânsız rotalar oluşturmaktadır. Küçük bir yüzdesi ise problemin tanımına uygun bir rota oluşturmaktadır. Alternatif rotaların (imkânsız ve eksik rotalar dâhil) tümü içerisinden ekranda 1 milisaniye aralıklarla o an hangi rotada kalındı ise o rota bilgisi görünmektedir. *Stop* butonuna basıldığında o an bulunan en küçük aylak zamanlı en iyi rota ekranda mesaj kutusu ile şekil 12'deki gibi gösterilmektedir. İşlemler yarıda kesildiğinden bulunan bu rota en iyi rota olmayabilir. Problem çözümü çok uzun değil ise hesaplamalar bittikten sonra yine mesaj kutusu ile ekranda en küçük aylak zamanlı en iyi rota ekranda gösterilmektedir.

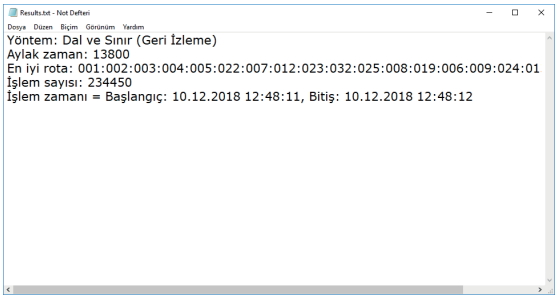
Şekil 12. Stop butonu ekran görünümü



Şekil 12'de rota dışında mesaj kutusunda işlem sayısı (her bir ileri ya da geri adım 1 işlem olacak şekilde) görülmektedir. Mesaj kutusunda yer alan İşlem zamanı kısmında ise işlemlerin başladığı tarih ve bittiği tarih görülmektedir. İşlem sayısı ortalama olarak saniyede 230.000 işlemdir.

Yukarıda yer alan Şekil 13'deki gibi *Go to Result* butonuna basıldığında ise sonuçlarla ilgili *Result.txt* metin belgesinde yer alan sonuçlar açılmaktadır. En son bulunan sonuç bu metin belgesine eklenmektedir. Sonuçlar aşağıda yer alan şekil 13'deki gibi görünmektedir.

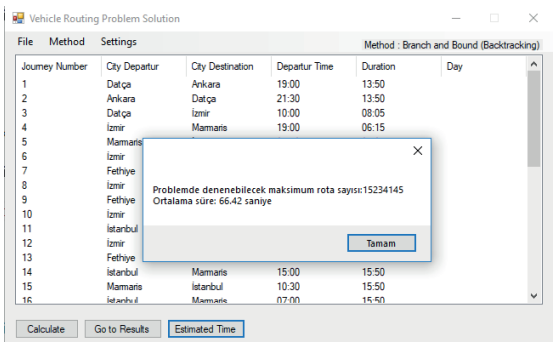
Şekil 13. Result.txt ekran görünümü



Her sonuç bulunduğunda önceki sonuç silinerek *Result.txt* dosyası değiştirilerek yenisi eklenmektedir.

Estimated Time butonuna bastığımızda ise problemin tahmini olarak ne kadar zamanda çözüleceği hesaplanıp gösterilmektedir. Bu bilgiler aşağıda yer alan şekil 14'deki gibi mesaj kutusunda gösterilmektedir. Buradaki tahmini süre bütün rotalar denendiğinde ortaya çıkabilecek en fazla süreyi göstermektedir. Gerçekte Geri İzleme Yaklaşımı sayesinde çoğu alternatif rota kötü çözüm üreteceğinden yarı yolda kesilip elenecektir. Böylece problemin çözümü tahmin edilen süreden daha kısa sürede bulunacaktır.

Şekil 14. Estimated time butonu ekran görünümü



3.4. Program dosyaları ve sistem gereksinimleri

Programın çalıştırıldığı bilgisayar konfigürasyonu aşağıdaki gibidir;

İşlemci : Intel Core i5 6200U 2.3GHz

Bellek : 4 GB

İşletim Sistemi : Windows 10 (64 bit)

Grafik Kartı : NVIDIA GeForce 940M

Program dosyaları aşağıda yer alan tablo 3'teki gibidir.

Tablo 3. Yazılıma ait dosyalar

Dosya Adı	**Açıklama**
VehicleRouting ProblemSolution.exe	Programın çalıştırıldığı exe dosyası
Veriler.accdb	Microsoft Access 2010 veri tabanı dosyası
Settings.txt	Veri tabanı dosyasının geçerli konumunun ayarlandığı metin belgesi dosyası
Result.txt	Sonuçlar için metin belgesi

Dosya büyüklükleri aşağıda yer alan tablo 4'teki gibidir.

Tablo 4. Yazılıma ait dosya büyüklükleri

Dosya Adı	**Dosya Büyüklüğü**
VehicleRoutingProblem Solution.exe	30 KB
Veriler.accdb	1.392 KB (Veri miktarı artarsa diskte kapladığı alan bir miktar daha artabilir)
Settings.txt	1 KB
Result.txt	1 KB

Programın bellekte kapladığı yer 8.2 MB civarıdır. İşlem yaparken 9.5 MB civarına çıkabilmektedir.

Program işlem yaparken işlemci performansının yaklaşık %30'unu kullanmaktadır.

Yukarıdaki konfigürasyona sahip bir bilgisayarda program her saniyede yaklaşık 230.000 adımı kontrol etmektedir.

4. SONUÇ

Araç Rotalama Problemleri genellikle mesafeyi enküçüklemeye yöneliktir. Bu çalışmada; mesafe yerine toplam aylak (boşta bekleme) zamanı enküçüklemeye yönelik bir problemin, optimali garanti eden bir yöntemle çözümü bulunmuştur. Literatürde henüz yeni tanımlanmış bir problem olan aylak zamanı en küçüklemeye yönelik Araç Rotalama Probleminin Geri İzleme yaklaşımı ile çözümüne yönelik yapılan bir çalışma olmadığı görülmüştür.

Program çalıştırıldıktan sonra 24 düğüm için hesaplamalar 1 dakika kadar sürmüştür. Düğüm sayısı çok büyük seçilmemiştir. Toplam aylak zaman 5715 dakika, en küçük aylak zamana ait rota; 1 ~~2 3 20~~ 21 10 11 12 ~~13~~ ~~14 15~~ 16 17 18 19 ~~22 23~~ 4 5 ~~6~~ 7 8 9 24 ~~ola~~rak ~~b~~ulu~~nm~~u~~ştu~~r. Bulunan sonuç kesin çözüm yöntemlerinden Geri İzleme Yaklaşımı ile çözüldüğünden bir en iyi (optimal) çözümdür.

KAYNAKÇA

BARKAOUI M., BERGER J. & BOUKHTOUTA A. (2015), Customer Satisfaction in Dynamic Vehicle Routing Problemwith Time Windows, *Applied Soft Computing*, Vol 35, ss.423-432.

BELL J. E. & MCMULLEN P. R. (2004). Ant Colony Optimization Techniques for the Vehicle Routing Problem, *Advanced Engineering Informatics*, C. 18, ss.41-48.

BORİSENKO A., HAIDL M. & GORLATCH S. (2017), A GPU Parallelization of Branch-and-Bound for Multiproduct Batch Plants Optimization, *Springer Science*, C. 73, ss.639-651.

BRUSCO M. J. & STAHL S. (2005), Statistics and Computing, Branch and Bound Applications in Combinatorial Data Analysis, *Springer Science*, ss.4-8.

CARIC T., CALIC A., FOSIN J., GOLD H. & REINHOLZ A., (2008). A Modelling and Optimization Framework for Real-World Vehicle Routing Problems, *Vehicle Routing Problem*, Intechopen.

DANTZİG G. B. & RAMSER J. H. (1959), The Truck Dispatching Problem, *Management Science*, 1959, C. 6, S. 2, ss. 80-91.

DASTGHAIBIFARD G.H., ANSARI E., SHEYKHALİSHAHİ S.M., BAVANDPOURİ A. & ASHOOR E. (2008), A Parallel Branch and Bound Algorithm for Vehicle Routing Problem, *Proceedings of the International MultiConference of Engineers and Computer Scientists*, C. 2, ss.1-6.

HOKAMA P., MIYAZAWA F. K. & XAVIER E. C. (2016), Abranch and cut approach for the vehicle routing problem with loading constraints, *Expert System With Applications*, C. 47, ss. 1-13.

KADRİ A. A., KACEM I. & LABADİ K. (2016), A Branch-and-Bound Algorithm for Solving the Static Rebalancing Problem in Bicycle-Sharing Systems, *Computers & Industrial Engineering*, C. 95, ss.45-52.

KESKİNTÜRK T., TOPUK N. & ÖZYEŞİL O. (2015), Araç Rotalama Problemleri ile Çözüm Yöntemlerinin Sınıflandırılması ve Bir Uygulama, İşletme Bilimi Dergisi, C. 3, S. 2, ss.77-107.

LAPORTE G. (1992), The Vehicle Routing Problem: An overview of exact and approximate algorithms, *European Journal of Operational Research*, C. 59, ss.345-358.

LİU S. B., NG K. M. & ONG H. L. (2008). Branch-And-Bound Algorithms For Simple Assembly Line Balancing Problem, *International Journal of Advanced Manufacturing Technology*, C. 36, ss.169-177.

McKEOWN G.P., RAYWARD-SMITH V.J. & TURPIN H.J. (1991), Branch-And-Bound as a Higher-Order Function, *Annals of Operations Research*, C. 33, ss.379-402.

MONTOYA A., GUÉRET C., MENDOZA J. E. & VILLEGAS J. G. (2016), A Multi-Space Sampling Heuristic for the Green Vehicle Routing Problem, *Transportation Research Part C*, C. 70, ss.113-128.

SEZEN H.K., (2017). *Yöneylem Araştırması*, Bursa: Dora Yayınevi,.