Gezici Personelin Otomatik Rota Planlamasının Web Tabanlı Programlama Araçları ile Gerçekleştirilmesi - Implementing Automatic Route Planning for Field Workers by using Web Based Programming Tools

İzzet Düdük, Onur Cambaz, Şükrü Ozan
AdresGezgini A.Ş. Ar-Ge Departmanı İzmir, Türkiye;
{izzetduduk,onurcambaz,sukruozan}@adresgezgini.com

Özet
Gezici personelin gün içerisinde gerçekleştirecekleri faaliyetlere ilişkin rota planlamasının, şirketin verimliliğini arttıracak şekilde optimize edilmesi önemli bir problemdir. Bu çalışmada ilgili problemin çözümüne yönelik olarak genetik algoritma tabanlı bir yöntem önerilmiştir. Mobilya nakliyesi ve kurulumu konusunda İzmir ili ve çevresinde hizmet veren bir firmanın müşteri veritabanı ve lojistik kaynaklarına ait bilgiler (personel, araç vb.) kullanılarak gerçekleştirilen simülasyonlar ile önerilen yöntemin bu problemin çözümünü geleneksel yöntemlere göre daha kısa sürede etkin bir şekilde gerçekleştirdiği gösterilmiştir. Ayrıca yöntem çevrimiçi sistemlerde çalışmasına olanak sağlayacak şekilde PHP web tabanlı yazılım dili ile gerçekleşmiş ve bu yöntemin geliştirilmekte olan personel takibi yazılımına entegre edilmesi sağlanmıştır.

Anahtar Kelimeler: genetik algoritma, otomatik rota planlaması, web tabanlı yazılım, PHP,MySQL

Abstract
It is an important problem to optimize the daily route planning of field workers in such a way to maximize the productivity of a company. In this study, a genetic algorithm based solution method for the corresponding problem is proposed.By using the results of the simulations which are iterated on the customer database and logistic source information of an İzmir based company which is serving for furniture transportation and installation, it's shown that the proposed method can effectively solve the problem effectively within considerably short period of time compared to conventional methods. Moreover the method is also implemented by using a web based programming language, i.e. PHP, such that it can also be run on online systems and the method is successfully integrated with a Personnel Attendance Control System which is currently in production phase.

Keywords: genetic algorithm, automatic route planning, web based programming, PHP, MySQL

1. GİRİŞ

Bir şirketteki gezici personelin en kısa sürede en fazla görevi tamamlayabileceği şekilde rota planlamasının yapılması literatürdeki Gezgin Satıcı Problemi ve bu problemle ilgili çözüm yöntemleri ile gerçekleştirilebilir[1]. Bekleneceği üzere problemin çözüm zorluğu personele atanacak görev sayısındaki artışlar paralelinde üstel olarak artmaktadır. Böyle bir problemi, özellikle personel ve görevlerden oluşan örneklem uzayının büyüklüğü arttığı takdirde kaba kuvvet yöntemleri (brute force methods) ile makul bir süre içerisinde çözmek pratikte mümkün olamamaktadır. Bu soruna optimum seviyede çözüm bulmak ve büyük zaman kayıplarının önüne geçmek için sezgisel yöntemler geliştirilmiştir[2].

Sezgisel algoritmalar doğal seçilim ilkelerine dayanan bir arama ve optimizasyon yöntemidir. Çözüm uzayının tamamını değil belirli bir kısmını tararlar. Böylece, etkin bir arama yaparak çok kısa bir sürede çözüme ulaşırlar. Diğer bir önemli üstünlükleri ise çözümlerden oluşan

popülasyonu eş zamanlı olarak incelemeleri ve böylelikle tek bir yerel en iyi çözüme takılmamalarıdır.

Proje konusu olan problem, en genel haliyle birden fazla personelin birden fazla göreve dağıtılması ile ilgilidir. Bu yapıdaki problemler Çoklu Gezgin Satıcı Problemi (Multiple Traveling Salesmen Problem) şeklinde tanımlanmaktadır ve Standart Gezgin Satıcı Problemine göre çok daha büyük bir çözüm karmaşıklığı içermektedir [3][4]. Literatürde en yaygın kullanılan sezgisel optimizasyon yöntemlerinden biri olan Genetik Algoritma yaklaşımının bu çalışma için de uygun olduğu görülmüş, ilk olarak çeşitli referans yayınlarında da kullanılmış olan bir MATLAB fonksiyonu [5] İzmir ili ve çevresinde, mobilya nakliyesi ve kurulumu konusunda hizmet veren bir firmanın müşteri veritabanı ve lojistik kaynaklarına ait bilgiler (personel, araç vb.) kullanılarak denenmiştir.

Problemin web tabanlı bir yazılım üzerine entegre edilecek bir modül ile çözümüne duyulan ihtiyaç dolayısı ile MATLAB dilinde yazılmış olan [5] çalışmasındaki algoritma bu çalışma kapsamında PHP dilinde yeniden uygulanmış ve MATLAB çıktısı ile aynı sonuçları verebilecek şekilde derlenmiştir. Yönteme eklenen bir erken durdurma koşulu (early stopping condition) ile yöntemin makul bir süre/döngü sayısı dahilinde en iyi sonucu vermesi sağlanmıştır. Bu da yöntemin web tabanlı gerçek zamanlı bir uygulama üzerinde kullanılabilmesini mümkün kılmıştır.

Bildirinin bir sonraki bölümünde gerçekleştirilen çalışmanın uygulama detayları verilecektir. Sonuç bölümünde elde edilen sonuçlar tartışılacak ve gelecek çalışmalara dair öngörüler sıralanacaktır.

2. UYGULAMA DETAYLARI

Uygulama aşamasında öncelikle ilgili problemin çözümüne yönelik yaygın olarak kullanılan önemli algoritmalar incelenmiştir. Açgözlü Algoritma (Greedy Algorithm) her personelin kendine en yakın olan görevi seçerek ilerlemesiyle çalışmaktadır. Dolayısıyla başladığı noktaya dönüşü hesaba katmamaktadır, aynı zamanda konumları birbirlerine yakın olan personelin aynı bölgedeki görevleri seçme ihtimalleri de vardır. Bu durum da bölgesel süre kayıplarına ve karmaşıklığa neden olmaktadır.

A* Algoritması [6] en kısa yol problemlerine çözüm üretmek için kullanılan Dijkstra Algoritmasının [7] farklı bir versiyonudur. Dijkstra Algoritması bir noktadan tüm noktalara olan uzaklıkları dikkate alarak en kısa yolu bulmaya çalışır. A* Algoritması ise hedefe ulaşmak için gerçek mesafenin üzerine hedefe olan sezgisel mesafeyi de ekleyerek her adımda bağlantılı olduğu tüm görev noktalarını kontrol ederek hesaplar. Gezici personelin başladığı noktaya geri dönmesi gerektiğinden dolayı A* Algoritmasının bu aşamada sezgisel mesafeyi dönüş için de hesaplaması gerekmektedir. Aynı zamanda bu çalışmada ele alınan çoklu personelin varlığından dolayı A* Algoritmasındaki hesaplamaların personel sayısı ile orantılı olarak hesaplama yükünün artması çözüm süresinin uzamasına neden olmaktadır. Bununla birlikte çalışmadaki görev sayısının artması A* Algoritması için alternatif çözüm uzayındaki denemelerin de artması anlamına gelmektedir. Bu da sistemin cevap verme süresinin uzamasına neden olmaktadır.

Zaman karmaşıklığı analizi, bir algoritmanın büyüme davranışını tahmin etmek için kullanılabilir ve algoritmanın gerçek zaman verimliliğini analiz etmek ve optimize etmek için kullanışlıdır. A* Algoritması zaman karmaşıklığı incelendiğinde en kötü ihtimalle genişletilmiş tüm noktalarda çözüm derinliği *d* olarak tanımlanır ve erişilebilecek noktaların sayısını temsil etmektedir. Her

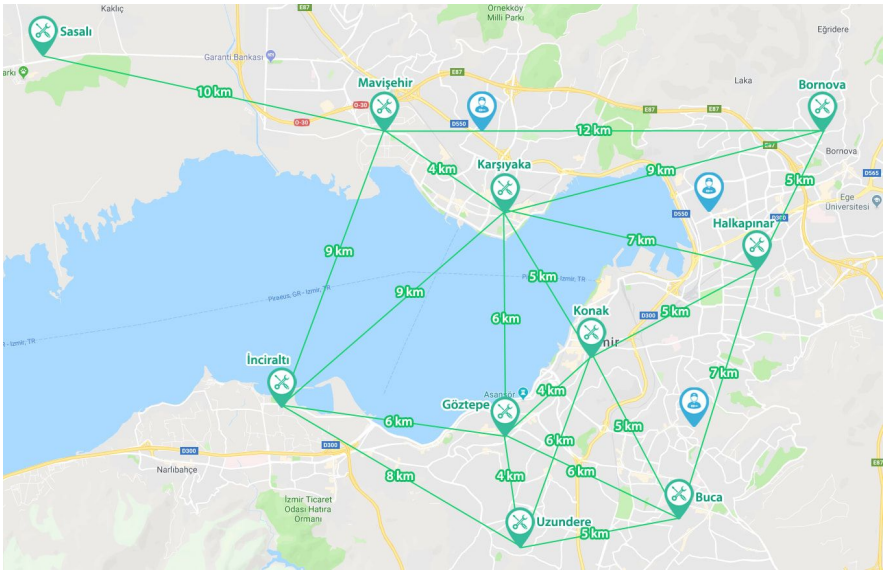
bir noktada bulunma durumu da dallanma faktörü olarak *b* ile temsil edilmektedir. Bu durumda arama ağacının dallarının her bir seviyede üstel olarak üretildiği bilinmektedir. Zaman karmaşıklığı da $O(b^d)$ sonucunu verir.

```
Sonuç: En Az Süre
minTime = INF (Varsayılan minimum süre);
popSize (Popülasyon büyüklüğü);
salesMan (Personel sayısı);
totalTime (Toplam Süre);
while Makul En uygun çözümü bulana kadar do
  for popSize do
    for salesMan do
      totalTime+=göreve uzaklık;
    end
  end
  if totalTime < minTime then
    minTime:=totalTime;
  end
  for p:= 8 TO popSize STEP 8 do
    bestOf8Route = En iyi 8'li kromozom;
    bestOf8Route = En iyi kırılım noktaları;
    I, Kromozom diziliminde rastgele bir indis;
    J, Kromozom diziliminde rastgele bir indis;
    for k:= 1 TO 8 do
      tmpRoute[k] = bestOf8Route;
      tmpBreak[k] = bestOf8Break;
      if 2 then
        tmpRoute[k]:=FLIP(I,J);
      end
      if 3 then
        tmpRoute[k]:=SWAP(I,J);
      end
      if 4 then
        tmpRoute[k]:=SLIDE(I,J);
      end
      if 5 then
        tmpBreak[k]:=RAND();
      end
      if 6 then
        tmpRoute[k]:=FLIP(I,J);
        tmpBreak[k]:=RAND();
      end
      if 7 then
        tmpRoute[k]:=SWAP(I,J);
        tmpBreak[k]:=RAND();
      end
      if 8 then
        tmpRoute[k]:=SLIDE(I,J);
        tmpBreak[k]:=RAND();
      end
    end
    newRoute[p]:=tmpRoute;
    newBreak[p]:=tmpBreak;
  end
  Route:=newRoute;
  Break:=newBreak;
end
```

Algoritma 1. Çalışmada kullanılan algoritmaya ait sözde program (pseudocode).

Yukarıda sıralanmış olan sebeplerden dolayı ilgili Çoklu Gezgin Satıcı Probleminin (Multiple Travelling Salesman Problem) çözümü için genetik algoritma tercih edilmiştir. Genetik algoritma ile elde edilen sonuçlara göre sistemde bulunan görevlerin önceden belirlenmiş kurallara göre en uygun personele atanması probleminde optimal çözüme en yakın çözüme klasik metotlara göre çok daha hızlı ve etkin bir şekilde ulaşıldığı ve yukarıda bahsedilen geleneksel yöntemlere göre minimum %30 oranında zaman tasarrufu sağladığı gözlemlenmiştir. Bu çalışmada kullanılan genetik algoritmada [5] çalışmasında önerilen algoritmadan esinlenilmiştir. MATLAB dilinde yazılan bu betiğin sözde kodu Algoritma 1 şeklinde gösterilebilir. Görüleceği üzere ilgili yöntem, optimizasyon için farklı kombinasyonları denemek adına klasik genetik algoritma bileşenleri olan ve doğal seçilimi taklit eden bir dizi işlemi her döngüde gerçekleştirmektedir. Fonksiyon içerisindeki kromozom yapılarının genetik algoritmanın doğası gereği yeni bireylerin oluşumu için gereken çaprazlama ve mutasyon işlemleri uygulanmıştır [8]. Bu işlemler sonucunda yeni bireylerin uygunluk değerleri hesaplanmış ve en optimal süre elde edilene kadar döngü sürdürülmüştür. Uygunluk değeri o popülasyonun verimini gösterir ve sonraki nesilleri temsil etme oranı daha yüksektir [9].

Çalışmada ilgili algoritmanın en önemli girdisi tüm görevlerin birbirlerine olan mesafe sürelerinin matrisidir. Önceki bölümlerde de bahsedildiği üzere bu çalışmada kullanılan veri İzmir ili ve çevresinde mobilya nakliyesi ve kurulumu konusunda hizmet veren bir firmaya ait gerçek verilerdir. Örnek bir mesai gününün hemen başında, firmanın günlük hizmetlerini vereceği müşterilerinin konumlarını Şekil 1 şeklinde, müşteriler arasındaki mesafeleri de kabaca kuş uçuşu mantığı ile gösterip, bu problemin çözümü için bir yaklaşım önermemizin mümkün olmadığını gözlemledik.



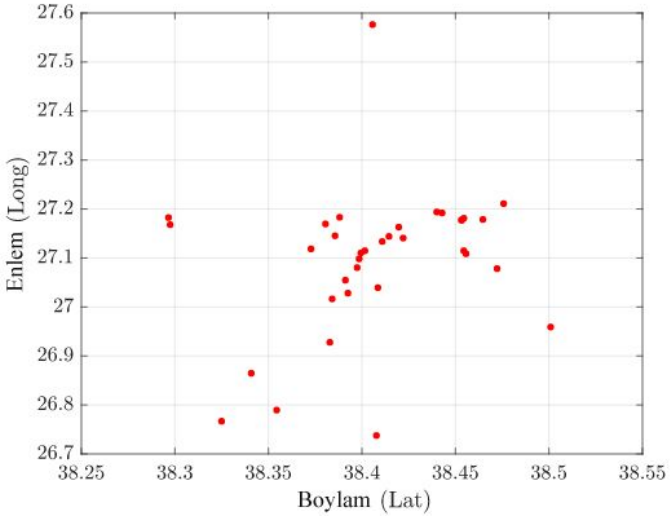
Şekil 1. Müşteri adreslerinin arasındaki mesafelerin İzmir ili haritası üzerinde kuş uçuşu ile gösterimi.

İzmir ili bir körfez etrafında kurulu olduğu için morfolojisi Şekil 1 üzerinde gösterilen bir graf topografisi için uygun değildir. Bunun yanında şehrin morfolojisi uygun olsa bile konumlar arasını kuş uçuşu mesafe ile göstermek özellikle İzmir gibi metropollerde problemin çözümü için yeterli olmamaktadır. Gün içerisindeki trafik yoğunluğu ve araçların kullanabileceği yolların uygunluk durumuna göre şehir üzerindeki iki nokta arasındaki mesafeyi katetmek için geçecek olan süre önemli ölçülerde değişiklik arz etmektedir.

İki nokta arasındaki yol durumunu (müsaitlik, trafik yoğunluğu v.b.) öğrenmek ve geçecek süreyi tahmin etmek amacıyla Google Maps Distance Matrix API servisi [10] kullanılmıştır. İlgili servis çıktı sonuçlarını JSON formatında verdiği için bu bilgi özellikle PHP gibi web tabanlı yazılım programlama dilleri ile kolayca okunup manipüle edilebilmektedir.

Google Maps API ile müşteri adresinden elde edilen tekil lokasyon bilgileri de enlem ve boylam bilgisi olarak MySQL veri tabanı üzerinde tutulmaktadır. Şekil 2, örnek bir günde İzmir şehrindeki hizmet lokasyonlarını MATLAB figürü olarak göstermektedir. Google Distance Matrix API kullanılarak harita üzerindeki herhangi iki nokta arasındaki araba ile ulaşım süresi biçiminden alınan verilerle bir uzaklık matrisi olarak Tablo 1'deki gibi gösterilebilir.

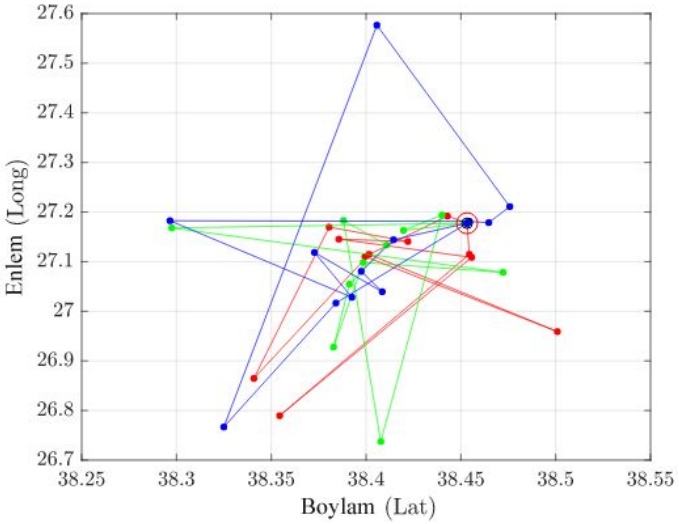
Şekil 3, Şekil 2'de gösterilen müşteri koordinatlarına göre elde edilen uzaklık matrisindeki süre cinsinden uzaklıklar ile hesaplanan en iyi rota sırasını 3 örnek personel için göstermektedir.



Şekil 2. Veri tabanındaki örnek bir güne ait hizmet lokasyonlarına ait enlem ve boylam ikililerinin MATLAB figürü ile gösterimi.

Tablo 1. Google Distance Matrix API kullanılarak oluşturulan süre bakımından uzaklık matrisi. Bu matris API'nin çıktısı olan JSON formatında alınıp işlenebilmektedir.

#	G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
G1:	0	26	20	16	14	9	11	22	20	0
G2:	25	0	16	23	26	28	31	40	40	25
G3:	18	17	0	10	20	22	24	35	33	18
G4:	14	22	12	0	20	20	20	30	28	14
G5:	13	23	18	18	0	13	17	25	26	13
G6:	10	29	24	20	13	0	16	23	24	10
G7:	13	29	25	19	16	14	0	19	17	12
G8:	23	40	35	30	25	25	20	0	17	23
G9:	22	40	36	30	28	25	20	17	0	22
G10:	0	23	20	16	14	9	11	22	20	0



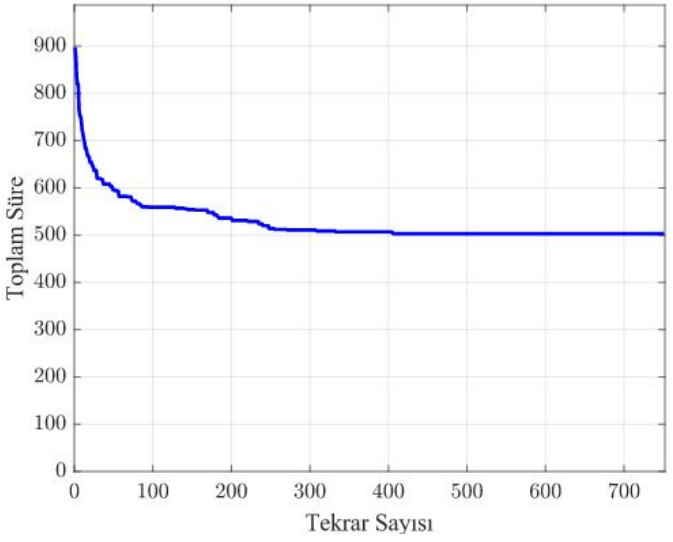
Şekil 3. Önerilen algoritmanın Şekil 2'de gösterilen koordinatlar için 3 personel için ürettiği rotalar. Mavi, yeşil ve kırmızı çizgilerin herbiri farklı bir personeli göstermektedir.

Şekil 2'de 3 örnek personel için hesaplanarak gösterilen rotaların harita üzerindeki gösterimleri de Şekil 4'teki gibi olmaktadır.

Şekil 4. Şekil 3'te 3 farklı gezici personel için sıralaması gösterilen rota planlaması sonuçlarının her bir personel için harita üzerindeki gösterimi.

İlgili yöntemin eniyileme süreci orjinal MATLAB kodu içerisinde döngü sayısına bir üst sınır verilerek sınırlandırılmıştır. Yaklaşık 800 döngü için gerçekleştirilen eniyileme süreci sonucunda hesaplanan toplam süre verisi Şekil 5'te görülebilir. Şekilden de görülebileceği üzere yaklaşık 400. döngüden sonra eniyileme sürecinde belirgin bir değişiklik görülmemektedir.

Bu eğilim her zaman geçerli olmaktadır. Problemin karmaşıklığı ile değişen bir süre boyunca eniyileme süreci devam etmekte ancak belli bir döngü sayısından sonra eniyileme sürecinde bir değişiklik olmamaktadır. Bu durumda döngüler için sabit bir üst sınır vermek bazı durumlar için yöntemin eniyileme süreci henüz bitmeden durmasına bazı durumlar için en iyi sonuca çok yakın bir sonuç elde edilmiş olmasına karşın döngülere devam etmeye sebep olmaktadır. Bu da pratikte gerçek zamanlı uygulamalarda bu haliyle yöntemin kullanımını zorlaştırmaktadır.



Şekil 5. Önerilen algoritmanın döngüsel eniyileme süreci.

Bu çalışmada [11] gibi güncel makine öğrenmesi çalışmalarında da sık sık karşımıza çıkan erken durdurma kriteri (early stopping criteria) metodolojisi uygulanmıştır. Kriter olarak 100

iterasyonda bir fonksiyon değerleri ile hesaplanan ve grafiğin o bölgedeki eğimi ile doğru orantılı olan bir sayı hesaplanmış ve bu sayının küçük sabit bir değerden küçük ya da eşit olduğu durumlarda eniyileme döngüsü sonlandırılmış ve programın farklı durumlar için ortalama çalışma süresi optimize edilmiştir.

MATLAB kodu üzerinde gerçekleştirilen çalışmalar sonrasında eniyileme sürecinin doğru bir şekilde gerçekleşip sürecin makul sürelerde sonlandığına emin olunan aşamadan sonra aynı algoritma bu sefer PHP programlama dili kullanılarak yeniden uygulanmıştır.

Özellikle Google'ın API servisleri ile elde edilen veriler MySQL veritabanı üzerine JSON verisi olarak kaydedilmiş ve daha alt düzeydeki verilerin MySQL sorguları içerisinde gerçekleştirilen json_extract metodu ile etkin bir şekilde okunması sağlanmıştır.

3. SONUÇ

Gerçekleştirilen bu çalışma AdresGezgini A.Ş. Ar-Ge Merkezi bünyesinde yürütülmekte olan 7170062 numaralı, "Web tabanlı düşük maliyetli ve gps destekli gezici personeli atayan denetleyen ve kontrol eden prototip sistemin geliştirilmesi" başlıklı TEYDEB 1507 projesi kapsamında gerçekleştirilmiştir.

Çalışmada firmaların sıklıkla karşılaştıkları gezici personelin rota planlaması problemi, web tabanlı bir sistem ile çözülmüştür. Öncelikle MATLAB kodu olarak geliştirilen algoritma örnek bir firmaya ait gerçek verilerle test edilmiş ve problem için uygun bir şekilde çalışır hale getirilmiştir. Algoritmanın çekirdeğini oluşturan genetik algoritmanın zaman karmaşıklığı incelendiğinde popülasyon sayısına kromozom uzunluğuna bağlı olarak değişiklik gösterdiği gözlemlenmiştir. Tüm popülasyon değerlendirilirse karmaşıklık $O(N\,L)$ (N, popülasyondaki birey sayısını , L ise kromozom uzunluğunu temsil eder) olmaktadır. Diğer durumlarda karmaşıklık kullanılmak istenilen operatörlere bağlı olarak da değişiklik gösterebilir. Yeni popülasyonların oluşturulmasında stokastik (rastlantısal) bir seçim uygulandığı için karmaşıklık $O(NlogN)$ olur. Yeni bireylerin içerisinden en iyilerin seçimiyle uygulanan turnuva metodunda ise karmaşıklık $O(N)$ olacaktır. Kısaca karmaşıklık kullanılmak istenen operatöre göre değişkenlik göstermektedir. Her durumda Big-O notasyonuna göre bu sonuçlar A* Algoritmasındaki $O(b^d)$ karmaşıklığından daha iyi sonuçlara sahiptir.

İlgili algoritma daha sonra PHP programlama dili kullanılarak yeniden uygulanmıştır. Bu sayede ilgili algoritma web tabanlı bir yazılım projesi olarak gerçekleştirilmekte olan personel takip yazılımı programına bir modül olarak eklenerek program için önem arz eden ana algoritmanın gerçeklenmesi sağlanmıştır.

Bundan sonraki çalışmalarda ilgili problemin farklı kısıtlar da eklenerek çözümüne yönelik olarak bir çalışma gerçekleştirilecektir. Örneğin gerek gerçekleştirilecek olan hizmetin gerekse müşterinin önceliği bu tarz saha çalışmalarında firmaların kullanmayı tercih ettikleri bir kriterdir. Bunların dışında ilgili bir personelin yetkinlik bilgisi, kullanılan araçların türü ve kapasitesi, personelin sorumlu oldukları alt coğrafi bölgeler gibi kısıtlar sistem parametresi olarak probleme eklenerek ilgili problem daha karmaşık ve çok boyutlu bir hale getirilecektir. Ayrıca daha kapsamlı ve gerçekçi bir sistem gün içerisinde aksaklıkların oluşması durumunda gerektiğinde rotaların yeniden hesaplanıp eniyilemesine olanak sağlamalıdır.

İlgili problemin karmaşıklığının çok fazla artması problemin burada bahsedilen geleneksel eniyileme yöntemlerinden ziyade günümüzde sıklıkla kullanılan yapay sinir ağı mimarileri ile çözülmesini gerekli kılacaktır. Bu konuda da araştırmalarımız ve çalışmalarımız devam edecektir.

4. REFERANSLAR

[1] T. Bektas, "The multiple traveling salesman problem: an overview offormulations and solution procedures,"Omega, vol. 34, no. 3, pp.209 – 219, 2006. [Online]. Available:http://www.sciencedirect.com/science/article/pii/S0305048304001550

[2] M. Yousefikhoshbakht, "An effective genetic algorithm for solvingthe multiple traveling salesman problem,"Journal of Optimization inIndustrial Engineering, vol. 8, pp. 73–79, 03 2011.

[3] K.-M. Lo, W.-Y. Yi, P.-K. Wong, K.-S. Leung, Y. Leung, and S.-T.Mak, "A genetic algorithm with new local operators for multipletraveling salesman problems,"International Journal of ComputationalIntelligence Systems, vol. 11, pp. 692–705, 2018/01. [Online]. Available:https://doi.org/10.2991/ijcis.11.1.53

[4] M. Nazarahari, E. Khanmirza, and S. Doostie, "Multi-objective multi-robot path planning in continuous environment using an enhancedgenetic algorithm,"Expert Systems with Applications, vol. 115, pp.106 – 120, 2019. [Online]. Available:http://www.sciencedirect.com/science/article/pii/S0957417418305165

[5] J. Kirk, "Fixed start/end point multiple traveling salesmen problem - genetic algorithm,"https://www.mathworks.com/matlabcentral/fileexchange/21299-fixed-start-end-point-multiple-traveling-salesmen-problem-genetic-algorithm, september 3, 2008.

[6] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristicdetermination of minimum cost paths,"IEEE Transactions on SystemsScience and Cybernetics, vol. 4, no. 2, pp. 100–107, July 1968.

[7] E. W. Dijkstra, "A note on two problems in connexion with graphs,"Numerische Mathematik, vol. 1, no. 1, pp. 269–271, Dec 1959.[Online]. Available:https://doi.org/10.1007/BF01386390

[8] J.-S. Jang, C.-T. Sun, and E. Mizutani, "Neuro-fuzzy and softcomputing-a computational approach to learning and machine intelli-gence [book review],"Automatic Control, IEEE Transactions on, vol. 42,pp. 1482 – 1484, 11 1997.

[9] Y. ¨O., "An overview of genetic algorithms,"Anadolu ¨Universitesi Bilimve Teknoloji Dergisi, vol. 2, p. 37–49, 2001.

[10] G. Inc., "Google distance matrix api, "https://developers.google.com/maps/documentation/distance-matrix/start, december, 2018.

[11] D. K. Duvenaud, D. Maclaurin, and R. P. Adams, "Early stopping asnonparametric variational inference," inProceedings of the 19th Inter-national Conference on Artificial Intelligence and Statistics, AISTATS2016, Cadiz, Spain, May 9-11, 2016, 2016, pp. 1070–1077. [Online].Available:http://jmlr.org/proceedings/papers/v51/duvenaud16.html