Fırat Üniversitesi Müh. Bil. Dergisi
34(2), 531-540, 2022

Araştırma Makalesi
https://doi.org/10.35234/fumbd.1077522

LabVIEW'de Kombinatoryal Arı Algoritması Araç Setinin Geliştirilmesi

Murat ŞAHİN*
Kontrol Sistemleri Geliştirme Müdürlüğü, Roketsan A.Ş., Ankara, Türkiye
* msahin@roketsan.com.tr

(Geliş/Received: 22/02/2022; Kabul/Accepted: 16/05/2022)

Öz: Bu çalışmada kombinatoryal problemler için LabVIEW'de geliştirilen Arı Algoritması (AA) Optimizasyon Araç Seti sunulmaktadır. LabVIEW, ölçüm ve kontrol uygulamalarında kullanılan, oldukça verimli bir geliştirme ortamıdır. Bu çalışma ise LabVIEW'un kombinatoryal optimizasyon bölümüne katkıda bulunmak amacıyla hazırlanmıştır. Bal arılarının polen bulma stratejisinden esinlenerek geliştirilen AA'nın tüm bölümleri, LabVIEW'de adım adım kodlanmıştır. Araç seti ile gezgin satıcı problemi üzerinde deneysel çalışmalar gerçekleştirilmiştir. Deneyler sonucunda 100 şehirlik problemlerde binde 3 ve daha küçük değerlerde, 150-200 şehirlik problemlerde ise %1.41'den daha küçük değerlerde sonuçlar elde edilmiştir. Ayrıca farklı optimizasyon algoritmaları ile gerçekleştirilen karşılaştırmalarda da daha iyi sonuçlar alındığı görülmüştür.

Anahtar kelimeler: Arı algoritması, LabVIEW, kombinatoryal optimizasyon, gezgin satıcı problemi.

Development of Combinatorial Bees Algorithm Toolkit in LabVIEW

Abstract: In this study, the Bees Algorithm (BA) Optimization Toolkit developed in LabVIEW for combinatorial problems is presented. LabVIEW is a highly productive development environment used in measurement and control applications. This study has been prepared to contribute to the combinatorial optimization part of LabVIEW. All parts of BA, inspired by the pollen searching strategy of honey bees, were coded step-by-step in LabVIEW. Experimental studies were carried out on the traveling salesman problem with the toolkit. As a result of the experiments, results were obtained at 3 per thousand and less in 100 city problems, and less than 1.41% in 150-200 city problems. In addition, it was observed that better results were obtained in comparisons with different optimization algorithms.

Key words: Bees algorithm, LabVIEW, combinatorial optimization, travelling salesman problem.

1. Giriş

Optimizasyon, verilen koşullar altında en iyi sonucu elde etme işlemidir ve özellikle mühendislik uygulamalarının önemli bir aşamasıdır. Mühendisler, herhangi bir mühendislik sisteminin tasarımında, yapımında, bakımında vb. farklı aşamalarda birçok teknolojik ve yönetsel kararlar almak zorundadır. Tüm bu kararların nihai amacı ya gerekli çabayı en aza indirgemek ya da kazancı en üst düzeye çıkarmaktır. Bu çaba veya hedeflenen kazanç, belirli karar değişkenlerinin bir fonksiyonu olarak ifade edilebiliyor ise, optimizasyon bu fonksiyonun maksimum veya minimum değerini veren koşulları bulma süreci olarak da tanımlanabilir [1]. Optimizasyon problemleri farklı yapılara sahip olabilmekle beraber, temelde sürekli ve kesikli olmak üzere ikiye ayrılır. Kombinatoryal problemler (KP) kesikli bir yapıya sahiptir ve popüler optimizasyon alanlarından biridir. Popüler olmasının sebeplerinden biri, orta ölçekli problemler için bile problemin çözümünün zor olmasıdır. Ayrıca, eğlence, turizm, lojistik, üretim gibi farklı alanlara uygulanabilmektedir. KPler arasında öne çıkanlar gezgin satıcı problemi (GSP) ve araç yönlendirme problemidir (AYP) [2]. GSP'de, satıcı bir rotadaki rastgele bir şehirden başlar ve diğer tüm şehirleri sadece bir kez ziyaret ederek ilk şehre geri döner. Amaç bu yolculuğu minimum maliyetle tamamlamaktır [3]. GSP'lerdeki olası çözümlerin sayısı doğrudan şehir sayısına bağlıdır ve (n-1)! ile ifade edilir [4]. Çok sayıdaki çözüm olasılığı, problemlerin optimizasyonu için araştırmacıları meta-sezgisel algoritmalara yöneltmiştir. Bu algoritmalar, topluluklardaki biyolojik süreçler, grup davranışı ve canlıların popülasyonda hayatta kalması vb. durumlardan esinlenerek geliştirilmiştir [5].

Popüler meta-sezgisel algoritmalardan birisi de Pham ve arkadaşları tarafından geliştirilen Arı Algoritmasıdır (AA) [6]. AA, gerçek bal arılarının polen arama davranışını taklit eder. Sadelik ve esneklik gibi avantajlarından dolayı mekanik tasarımdan enerji optimizasyonuna kadar birçok alanda kullanılmaktadır. KPler kapsamında ise, makine çizelgeleme problemleri [7], baskı devre montaj optimizasyonu [8], araç rotalama problemleri [9], tedarik zinciri optimizasyonu [10] ve GSP [11] gibi farklı alanlarda kullanılmıştır.

* Sorumlu yazar: msahin@roketsan.com.tr Yazarın ORCID Numarası: 0000-0002-3659-3528

AA'nın, C [12] ve Python [13] gibi programlama dillerinde kodlandığı ve kullanıldığı bilinmektedir. Endüstride popüler diğer bir program ise National Instrument firması tarafından geliştirilen LabVIEW (Laboratory Virtual Instrument Engineering Workbench) programıdır. Grafiksel programlama yöntemini kullanan güçlü bir araçtır [14]. Günümüzde özellikle kontrol, robotik ve otomasyon gibi alanlarda gerçek zamanlı uygulamalar kapsamında tercih edilmektedir. Bu alanlarda gerçekleştirilen optimizasyon çalışmaları kapsamında bünyesinde bulunan "Optimizasyon Araç Seti" kullanılmaktadır. Bu araç seti içerisinde, birinci ve ikinci dereceden problemlerin çözümü için hazırlanmış bloklar bulunmaktadır. Ayrıca sürekli fonksiyonların minimum değerlerini bulmak için bazı bloklar da mevcuttur [15]. Kompleks optimizasyon için ise, ENIT (LabVIEW'e araç seti geliştiren bir topluluk.) tarafından geliştirilen AMA (Advanced Metaheuristics Algorithms - Gelişmiş Meta-sezgisel Algoritmalar) araç seti bulunmaktadır. Bu araç seti bünyesinde; Parçacık Sürü Optimizasyonu, Karınca Kolonisi Algoritması, Yerçekimi Arama Algoritması, Öğretme-Öğrenme Tabanlı Optimizasyon ve Bozkurt Algoritması blokları bulunmaktadır [16]. Bunların haricinde araştırmacılar tarafından, optimizasyon çalışmaları kapsamında; Parçacık Sürü Optimizasyonu [17], Yarasa Algoritması [18], Karınca-Aslan Optimizasyonu [19] ve Bozkurt Algoritması [20] yöntemlerinin LabVIEW'de kodlandığı ve kullanıldığı görülmektedir. Literatür taraması yapıldığında, AA'nın LabVIEW'de kodlanmadığı görülmüş ve bu çalışmanın yapılmasına karar verilmiştir. Ayrıca, diğer optimizasyon çalışmaları ve araç setleri sürekli problemler üzerine geliştirilmiştir. Bu çalışmada kombinatoryal bir problem olan GSP ele alınmıştır. Çalışmanın ikinci bölümünde Kombinatoryal AA (KAA) hakkında bilgiler yer almaktadır. Üçüncü bölümde KAA araç seti ile ilgili detaylı bilgiler verilmiştir. Dördüncü bölümde araç seti ile gerçekleştirilen optimizasyon çalışmalarının sonuçları yer almaktadır. Araç seti NI LabVIEW 2019 (32-bit) versiyonunda hazırlanmıştır.

2. Kombinatoryal Arı Algoritması

KAA, diğer popülasyon tabanlı meta-sezgisel algoritmalarda olduğu gibi, optimizasyon problemini yinelemeli olarak çözer. Lokal ve global arama bölümleri vardır. Lokal arama bölümünde elit sitelerde daha fazla mahalle araması yapılırken diğer sitelerde daha az arama yapılır. Algoritma, kâşif arı sayısı (n), lokal arama sitesi sayısı (m), elit site sayısı (e), elit sitedeki toplayıcı arı sayısı (nep), diğer lokal sitedeki (nsp) toplayıcı arı sayısı, arama komşuluğunun büyüklüğü (ngh) ve yineleme sayısı (I) gibi bazı parametrelerin belirlenmesi ile başlar [21].

Kombinatoryal problemlerde arama bölümleri ve komşuluk tanımları sürekli problemlerden farklıdır. GSP'ler için, rota üzerindeki şehirlerin yerleri belirli kurallara göre konumları değiştirilerek daha az maliyetle yeni rotalar bulunmaya çalışılır. Arı algoritması için yapılan güncel çalışmalardan birinde Ismail ve arkadaşları [11], lokal arama bölümünde üç farklı operatör kullanmıştır: takas, ekleme ve tersine çevirme. Takas operatöründe, rastgele seçilen iki şehrin sırası değiştirilir. Ekleme operatöründe, rastgele seçilen bir şehir, rastgele seçilen farklı bir sıraya eklenir. Tersine çevirme operatöründe ise, rastgele seçilen iki şehir arasındaki sıra tersine çevrilir [11]. Bu operatörler, elit sitelerde daha fazla sayıda ve diğer lokal sitelerde daha az sayıda olmak üzere her döngüde rastgele kullanılır. Bu çalışmada, bu 3 operatöre ek olarak algoritmaya "değişken çoklu-ekleme" operatörü eklenmiştir [22]. Rastgele belirlenen bir bölgeden, birden çok ve değişken sayıda şehirler alınır. Bu şehirler komşulukta belirlenen ikinci bir konuma yerleştirilir. Bu sayede, bir adımda birden fazla şehrin yeri topluca değiştirilerek farklı bir deneme daha yapılmış olur. KAA'nın genel akış diyagramı ve elit site arama akış diyagramı Şekil 1'de verilmiştir.

3. Algoritmanın LabVIEW'de Geliştirilmesi

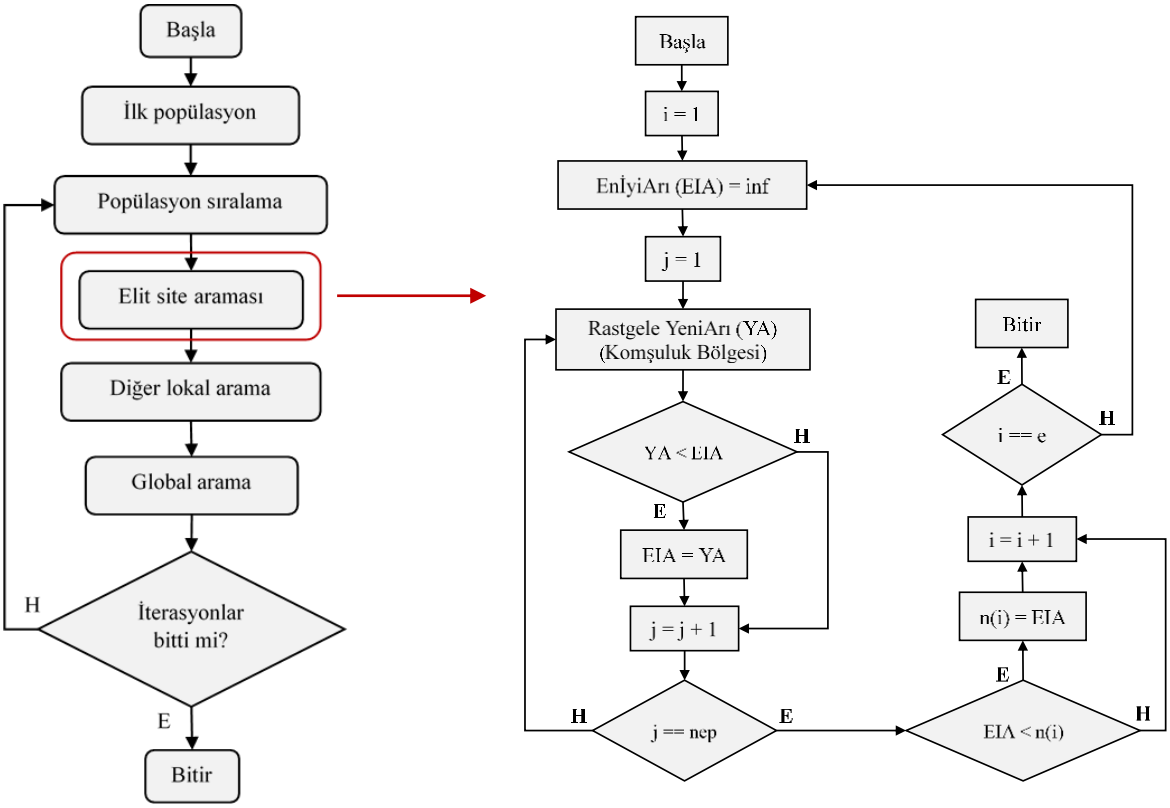
İlk olarak şehirler arası mesafelerden oluşan bir matris oluşturulmuştur. Algoritmada, her şehrin başka bir şehre olan uzaklığı bu matris kullanılarak hesaplanır. İki şehir arasındaki mesafe denklemi (1)'de verilmiştir [3].

$$d\big(T(i),T(i+1)\big)=\sqrt{(x_i-x_{i+1})^2+(y_i-y_{i+1})^2} \tag{1}$$

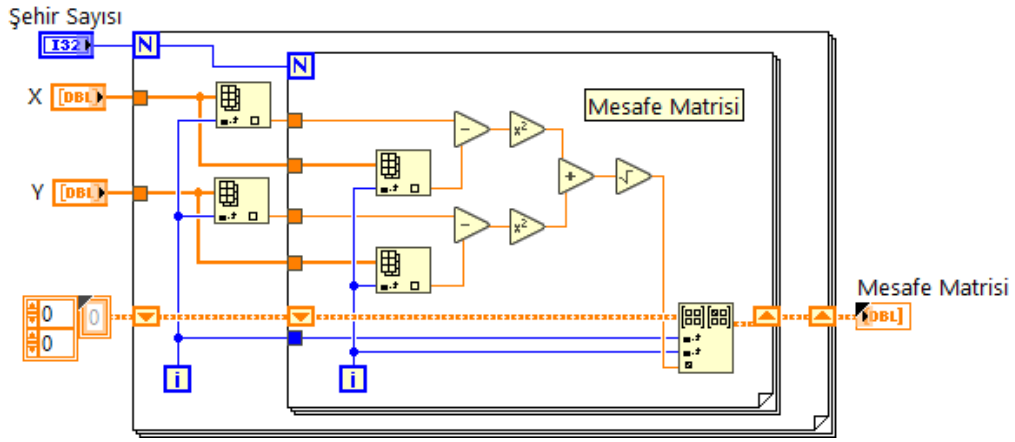
x ve y şehrin koordinatlarını gösterir. Denklem (2)'de ise toplam tur mesafesi gösterilmiştir [3].

$$\sum_{i=1}^{n-1} d\big(T(i),T(i+1)\big)+d\big(T(n),T(1)\big) \tag{2}$$

Mesafe matrisi Şekil 2'de gösterilmiştir. İçteki **for** döngüsünde (1) denklemi, dıştaki **for** döngüsünde ise (2) denklemi gerçekleştirilmektedir.



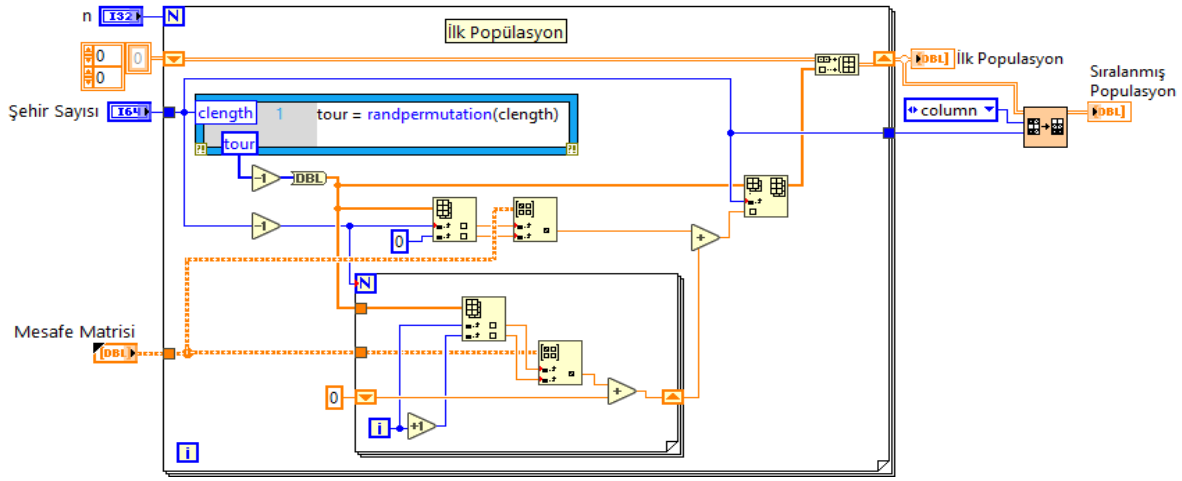
Şekil 1. KAA'nın genel akış diyagramı ve elit site arama



Şekil 2. Mesafe matrisi

3.1. İlk popülasyon

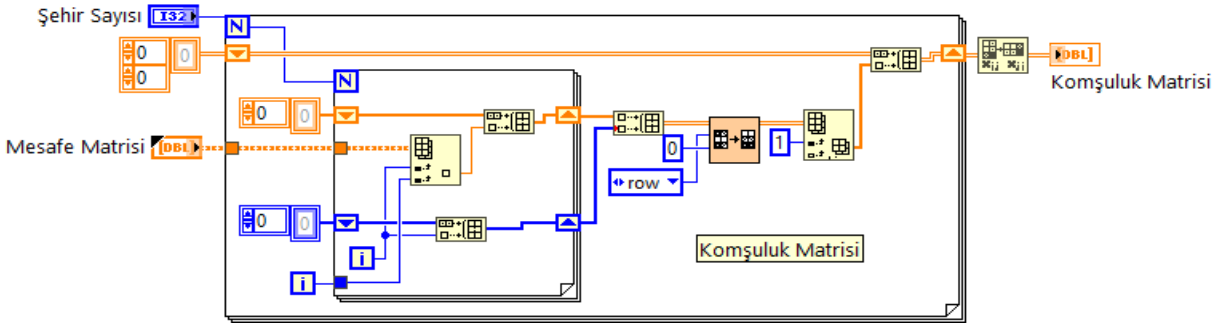
KAA'da ilk popülasyon, diğer birçok meta-sezgisel algoritmada olduğu gibi rastgele oluşturulur. Şekil 3'te popülasyon oluşumu gösterilmiştir. İlk olarak problemdeki şehirler rastgele sıralanır ve bir diziye aktarılır. Bu işlem formül bloğu içerisinde **randpermutation** komutu ile yapılır. Sonrasında **for** döngüsü içerisinde bu dizilimin maliyeti hesaplanır ve bu işlemler popülasyon sayısı **n** kadar tekrarlanır. Rota ve maliyeti aynı dizide tutulur. Dizinin son değeri maliyettir. Sonraki adımda ise popülasyonun üyeleri en düşük maliyetliden en yükseğe doğru olacak şekilde sıralanır. Bu işlem için **Sort 2D Array** bloğu kullanılır.



Şekil 3. İlk popülasyon

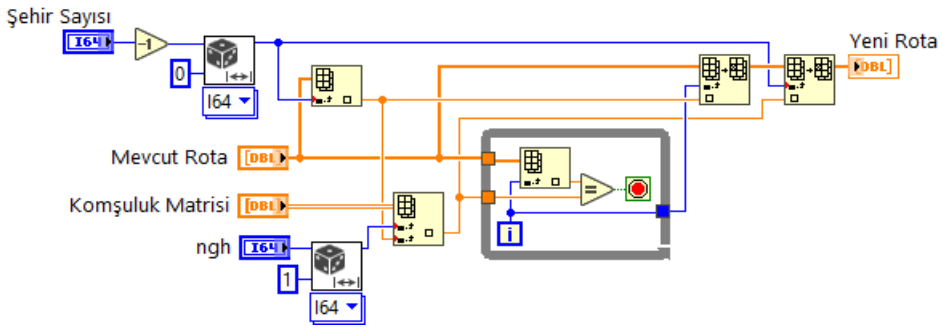
3.2. Lokal arama

Lokal aramada rota üzerindeki şehirlerin yerleri, önceki bölümde anlatılan operatörler ile değiştirilerek daha az maliyetle yeni rotalar bulunmaya çalışılır. İlk adımda, komşuların belirlenmesi için, her bir şehire yönelik komşuluk matrisi (Şekil 4) oluşturulmuştur. Her bir şehir için, diğer şehirlere olan uzaklıklarına göre yakından-uzağa mesafe sıralaması yapılır. Dıştaki **for** döngüsünde şehirler sıra ile alınır ve içteki **for** döngüsü ile diğer tüm şehirlere olan uzaklıkları bir dizide toplanır. Bu dizi sonrasında küçükten büyüğe sıralanarak yakın komşuluklar belirlenir. Bu işlem tüm şehirler için yapılır.



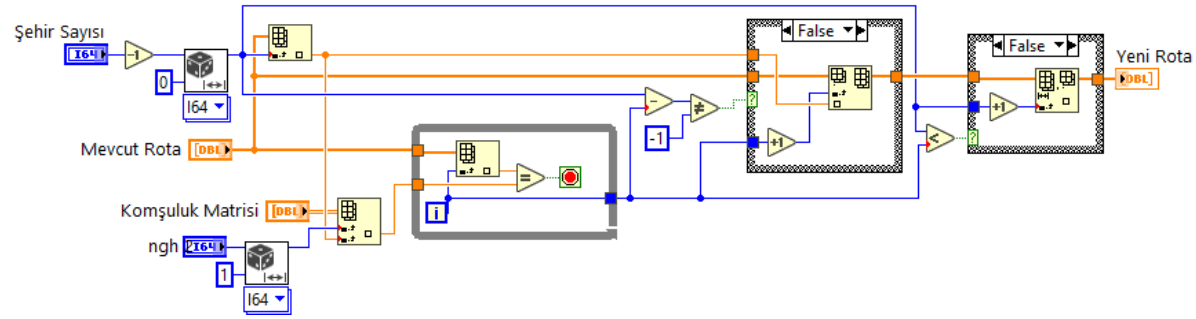
Şekil 4. Komşuluk matrisi

Takas operatöründe ilk şehir rastgele seçilir, diğer şehir ise komşular (**ngh**) arasından rastgele belirlenir. Burada ikinci seçimde aynı şehri seçmemek için **while** döngüsü kullanılır. Değiştirme işlemi **Replace Array** bloğu ile gerçekleştirilir. Takas operatörünün kodu Şekil 5'te gösterilmiştir.



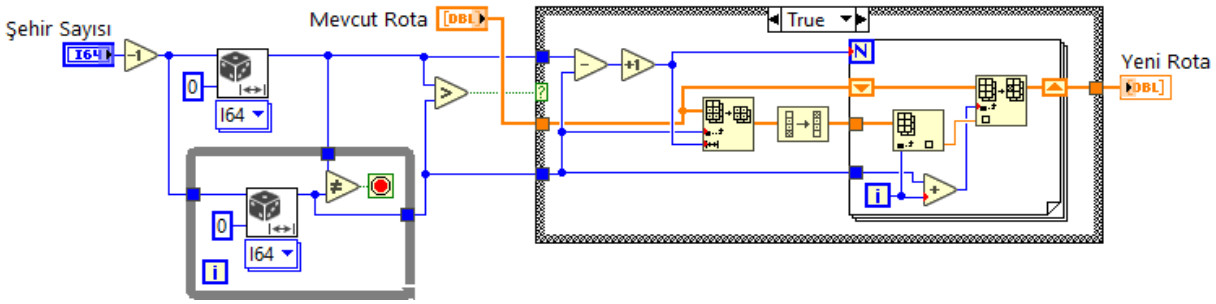
Şekil 5. Takas operatörü

Ekleme operatöründe, rastgele seçilen bir şehir, komşuluk içerisinde rastgele seçilen farklı bir sıraya eklenir. Aynı konumun seçilmemesi için burda da **while** döngüsü kullanılmıştır. "Ekleme" yapıldıktan sonra seçilen şehir hâlen ilk konumda durmaktadır, sonraki adımda rotadan silinir **(Delete From Array** bloğu ile.). Ekleme operatörü Şekil 6'da gösterilmiştir.



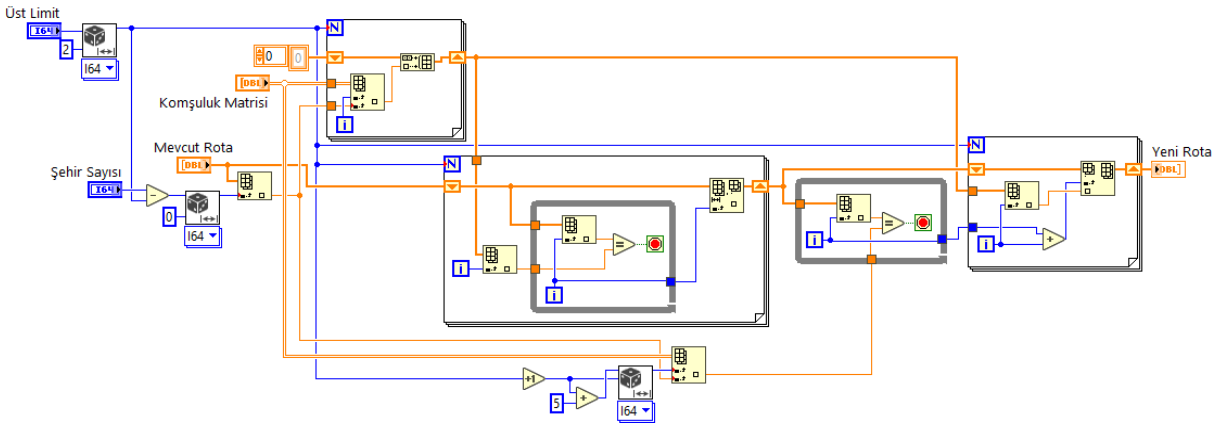
Şekil 6. Ekleme operatörü

Tersine çevirme operatöründe, rastgele seçilen iki şehir arasındaki sıra tersine çevrilir. Takasda olduğu gibi burada da iki farklı şehir belirlenir. Belirlenen konumlar arasındaki şehirlerin sıralaması dizi işlemleri ile tersine çevrilmektedir (Şekil 7). İlk olarak **Array Subset** bloğu ile ilgili bölge alınır ve bir for döngüsü içerisinde sıralaması tersine çevrilerek yeniden aynı rota içerisine kayıt edilir.



Şekil 7. Tersine çevirme operatörü

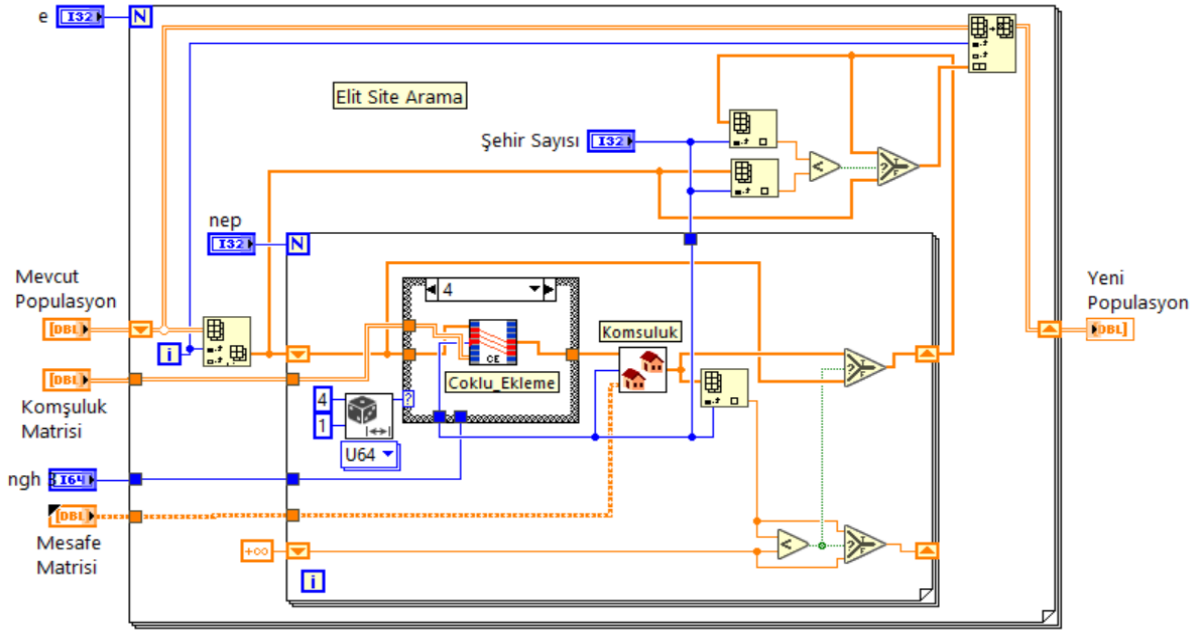
Çoklu ekleme operatöründe, ilk olarak seçilecek şehir sayısı ve şehirlerin alınacağı konum belirlenir. Bu konumdan belirlenen sayı kadar şehir seçilir. Daha sonra, rotada eklenecek olan ikinci konum belirlenir. Çoklu ekleme operatörü Şekil 8'de gösterilmiştir.



Şekil 8. Çoklu ekleme operatörü

Bu operatörler, elit sitelerde daha fazla sayıda ve diğer lokal arama bölümünde daha az sayıda olmak üzere her döngüde rastgele kullanılır. Elit site araması Şekil 9'da gösterilmiştir. İçteki **for** döngüsünde, elit sitede her bir

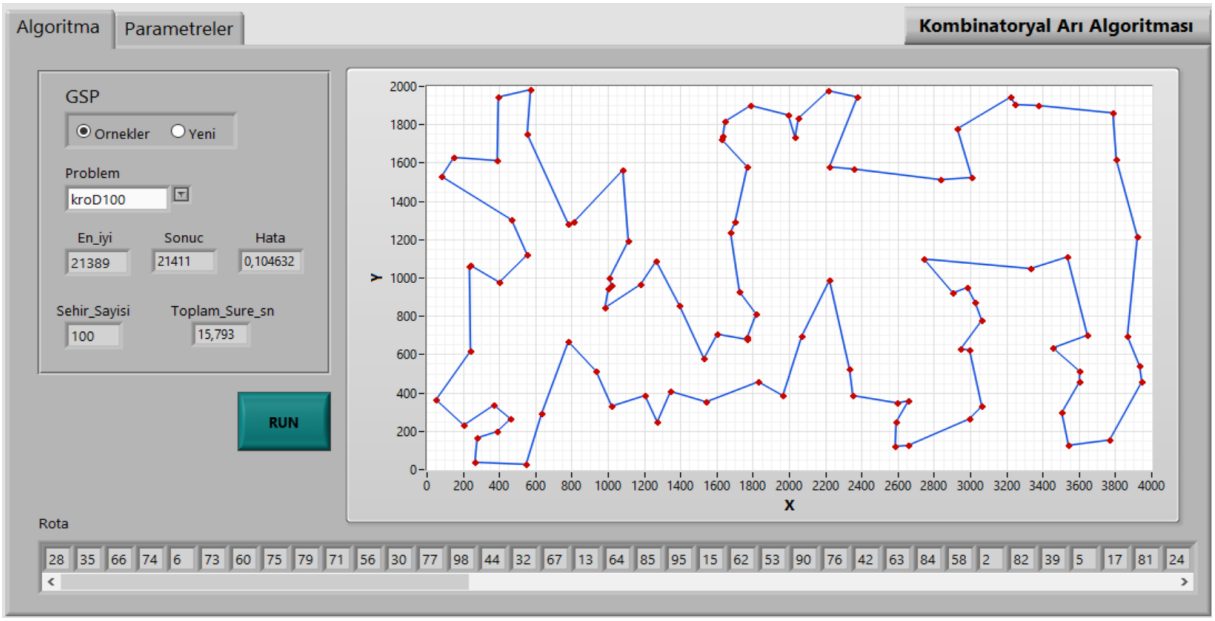
rota için "nep" sayısı kadar arama yapılmakta ve bunlardan en iyisi seçilmektedir (**Select** bloğu ile.). Dıştaki **for** döngüsünde ise bulunan bu değerler popülasyondaki mevcut değer ile karşılaştırılmakta ve daha iyi ise mevcut değerin yerine kayıt edilmektedir. Şekil 9'da gösterilen arama bölümü elit olmayan diğer lokal sitelerde de aynen kullanılmaktadır. Sadece e yerine "m" ve nep yerine "nsp" sayıları gelmektedir. Global arama bölümü de ilk popülasyon oluşturma bölümü ile aynıdır, oluşturulan yeni rastgele rotalar lokal arama dışında kalan bölgelere eklenir.



Şekil 9. Elit site araması

4. Deneysel Çalışmalar

Programın kullanıcı arayüzü Şekil 10'da verilmiştir. Bu bölümde bazı örnek GSP'ler mevcuttur. Ayrıca "Yeni" ifadesi tıklanarak X-Y koordinatlarını içeren farklı bir örnek seçilebilir.



Şekil 10. Kullanıcı arayüzü

Deneysel çalışmalar Intel Core i5-8265U CPU 1.60 GHz, 8 GB RAM ve Windows 10 Pro'ya (64 bit işletim sistemi) sahip bir Lenovo Thinkpad bilgisayar üzerinde gerçekleştirilmiştir. Bu çalışmada kullanılan tüm problemlerin verileri MP-TESTDATA'dan alınmıştır [23].

Algoritma parametreleri belirlenirken önceki çalışmalar referans alınmıştır [24,25]. Popülasyon sayılarının 20 ile 100 arasında değiştiği görülmektedir. Popülasyon sayısının fazla olması işlem yükünün artmasına neden olduğu görülmektedir. Bu nedenle popülasyon sayısı (n) 40 olarak belirlenmiştir. Lokal site sayısı (m) popülasyonun yarısı (20) olarak belirlenmiştir. Elit siteler (e) için ise "m"nin %25'i (5) belirlenmiştir. Algoritmada kullanılan tüm parametreler Tablo 1'de verilmiştir.

Tablo 1. Algoritma parametreleri

Parametre	**Değer**
Tekrar Sayısı	10
İterasyon Sayısı (I)	1000
Popülasyon Sayısı (n)	40
Elit Sitedeki Arı Sayısı (nep)	300
Diğer Lokal Sitelerdeki Arı Sayısı (nsp)	100
Elit Site Sayısı (e)	5
Lokal Site Sayısı (m)	20

Algoritmayı test etmek için MP-TESTDATA'dan 10 farklı GSP seçilmiştir. Deneysel sonuçlar Tablo 2'de gösterilmiştir. Tablonun ilk sütunu GSP'nin adını ve bilinen en iyi değerini göstermektedir. Sonraki beş sütun bu çalışmanın sonuçlarını göstermektedir. İlk iki sütun, 10 yinelemedeki en iyi değeri (Min) ve 10 değerin ortalamasını (Ort) gösterir. %min ifadesi, minimum değerin bilinen en iyi değerden sapmasını verir. %ort ise, 10 çalıştırma sonucunda bulunan elde edilen ortalama değerin bilinen en iyi değerden sapma oranını gösterir. Son sütun ise her iterasyonda en iyi değerin bulunduğu süreyi gösterir. %min ve %ort denklemleri (3) ve (4)'te tanımlanmıştır [12].

$$\%ort = \frac{(Ort - En\ İyi)}{(En\ İyi)} * 100\% \tag{3}$$

$$\%min = \frac{(Min - En\ İyi)}{(En\ İyi)} * 100\% \tag{4}$$

Tablo 2. Deneysel sonuçlar

GSP (En İyi)	**Min**	**Ort**	**%min**	**%ort**	**Ort. Süre (s)**
Berlin52 (7542)	7544	7544	0.03	0.03	10.24
St70 (675)	677	679.4	0.3	0.65	17.64
Pr76 (108159)	108159	108350	0	0.18	17.59
KroA100 (21282)	21285	21289.2	0.01	0.03	16.74
KroB100 (22141)	22139	22233.3	0	0.42	21.48
Ch150 (6528)	6620	6666	1.41	2.11	19.04
KroA150 (26524)	26761	26971.1	0.89	1.68	21.42
KroB150 (26130)	26320	26473	0.73	1.31	21.83
KroA200 (29368)	29644	30038.8	0.94	2.28	24.88
Lin318 (42029)	44554	44989.4	6.01	7.04	37.71

Tablo 2'den de görüleceği gibi, KAA ile 52-100 arası şehre sahip GSPler için bilinen en iyi değerler veya binde/on binde sapma ile yakın değerler bulunmuştur. Bu problemlerde ortalama değerlerdeki sapmalar da %1'in altında olmuştur. 150 - 200 şehirlik problemler için ise bilinen en iyi değerlerden %0,73 - 1,41 arasında sapmalarla

sonuçlar elde edilmiştir. Bu problemler için 10 testin ortalamasındaki sapmalar ise %1.31-2.28 arasındadır. KAA araç setinin GSPler kapsamında başarıyla çözüm üretebildiği görülmektedir. Bazı problemler için bulunan en iyi rotalar Ek'te gösterilmiştir. KAA araç setinin etkinliğini değerlendirebilmek için ayrıca literatürde bulunan farklı çalışmalar ile karşılaştırma yapılmıştır. Bu kapsamda Hibrit Balina Optimizasyonu [26], Genetik Algoritma [27] ve En Yakın Komşu Algoritması [27] seçilmiştir. Karşılaştırma sonuçları Tablo 3'te gösterilmiştir.

Tablo 3. Diğer algoritmalar ile karşılaştırma sonuçları

GSP (En İyi)	KAA		HBO [26]		GA [27]		EYK [27]	
	Min	Ort	Min	Ort	Min	Ort	Min	Ort
Berlin52 (7542)	**7544**	**7544**	7868	7971	**7544**	7836.7	7713	7887.2
St70 (675)	**677**	**679.4**	766	789.6	**677**	694.3	691	711.6
Pr76 (108159)	**108159**	**108350**	132654	134651	**108159**	109675	108880	113480
KroA100 (21282)	**21285**	**21289.2**	24390	25306.2	21307	21754.9	21395	21551.2
KroB100 (22141)	**22139**	**22233.3**	-	-	22338	22615.3	22347	22678
Ch150 (6528)	**6620**	**6666**	-	-	6627	6800.4	6622.8	6667.7
KroA150 (26524)	**26761**	**26971.1**	31798	31948.7	27080	27897.5	27354	27936.2
KroA200 (29368)	**29644**	**30038.8**	-	-	30009	30899.6	29907	30286.1
Lin318 (42029)	44554	44989.4	-	-	46186	47514	**43743**	**44236.5**

Tablo 3.'ten görüleceği üzere, KAA araç seti ile elde edilen sonuçlar diğer çalışmaların sonuçlarına göre daha başarılı olmuştur. Özellikle ortalama değerlerde, diğer algoritmalardan çok daha düşük maliyetli rotalar hesaplanmıştır. Bu araç setinin tekrarlanabilirlik açısından güçlü olduğunu göstermektedir. Diğer algoritmalardan sadece EYK 318 şehirlik problemde daha iyi rotalar hesaplayabilmiştir.

5. Sonuç

Bu makalede, kombinatoryal problemlerin optimizasyonu için LabVIEW'de geliştirilen Arı Algoritması Araç Seti anlatılmıştır. Çalışmalar adım adım anlatılarak okuyucunun kendi uygulamalarını geliştirmesine fırsat verilmektedir. Lokal arama bölümünde 4 farklı arama operatörü gösterilmiştir. Farklı uygulamalar için bu bölüme yeni operatörler kolayca eklenebilir. Makalede örnek olarak GSP üzerine çalışılmış olmasına rağmen, lokal arama bölümü farklı kombinatoryal problemler için de kullanılabilir. Yeni problemler ve operatörler tanımlanırken LabVIEW'in çalışma prensibi göz önünde bulundurulmalıdır. Programlama dillerinde yukarıdan aşağıya sıralı bir akış söz konusuyken, LabVIEW kodları paralel bir şekilde çalıştırır. Dolayısıyla sıralı işlem yapılması gerektiğinde "Sequence" (sıra) blokları tercih edilmelidir. Araç seti ile yapılan deneylerin sonucunda, 100 şehre kadar olan problemlerde %1'den daha düşük sapma değerleri ile ve 150-200 şehirlik problemler için de %1-2 aralığında sapma değerleri ile sonuçlar elde edilmiştir. Literatürde bulunan bazı çalışmalar ile karşılaştırıldığında daha iyi sonuçların alındığı da görülmüştür. Buna rağmen ileriki çalışmalarda, 300 üzeri şehre sahip problemler için araç setinde iyileştirme çalışmaları gerçekleştirilebilir ve ayrıca farklı kombinatoryal problemler üzerinde de denenebilir.

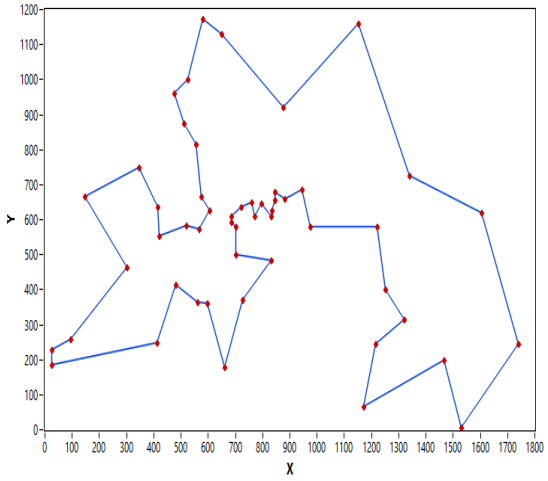
Teşekkür

Yazar, bu çalışmaya vermiş olduğu maddi destekten dolayı, Roketsan A. Ş.'ye teşekkürlerini sunar.

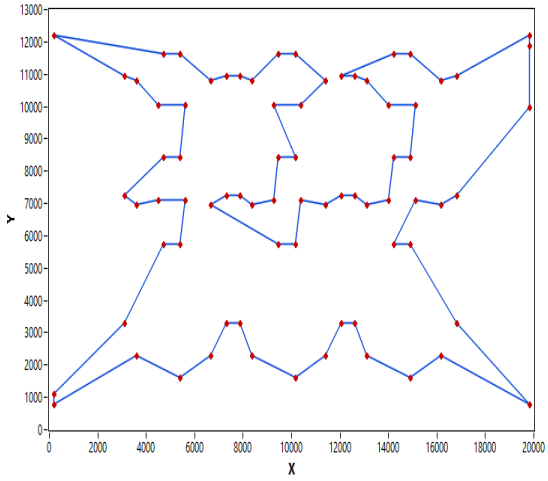
Kaynaklar

[1] Rao S S, Engineering Optimization Theory and Practice. John Wiley & Sons, New Jersey, USA, 2009.
[2] Osaba E, Yang X, Ser JD. Traveling salesman problem: a perspective review of recent research and new results with bio-inspired metaheuristics. Nature-Inspired Computation and Swarm Intelligence Algorithms, Theory and Applications 2020, 135-164.
[3] Karaboga D, Gorkemli B. Solving Traveling Salesman Problem by Using Combinatorial Artificial Bee Colony Algorithms. International Journal on Artificial Intelligence Tools 2019, 28(1). 1950004 (28 pages).
[4] Chen SM, Chien CY. Parallelized genetic ant colony systems for solving the traveling salesman problem. Expert Systems with Applications 2011, 38: 3873–3883.
[5] Hamzadayi A, Baykasoglu A, Akpinar S. Solving combinatorial optimization problems with single seekers society algorithm. Knowledge-Based Systems 2020, 201–202.
[6] Pham DT, Castellani M. The Bees Algorithm: Modelling Foraging Behaviour to Solve Continuous Optimization Problems. Proceedings of the Institution of Mechanical Engineers, Part C, Journal of Mechanical Engineering Science 2009, 223(12): 2919-2938.
[7] Packianather MS, Yuce B, Mastrocinque E, Fruggiero F, Pham DT, Lambiase A. Novel Genetic Bees Algorithm applied to single machine scheduling problem. World Automation Congress (WAC) 2014, 1-6.
[8] Ang MC, Pham DT, Ng KW. Application of the Bees Algorithm with TRIZ-inspired operators for PCB assembly planning. Proceedings of 5 th Virtual International Conference on Intelligent Production Machines and Systems 2009, 454-459.
[9] Alzaqebah M, Jawarneh S, Sarim HM, Abdullah S. Bees Algorithm for Vehicle Routing Problems with Time Windows. International Journal of Machine Learning and Computing 2018, 8(3): 236-240.
[10] Yuce B, Mastrocinque E, Lambiase A, Packianather MS, Pham DT. A multi-objective supply chain optimisation using enhanced Bees Algorithm with adaptive neighbourhood search and site abandonment strategy. Swarm and Evolutionary Computation Volume 2014, 18: 71-82.
[11] Ismail AH, Hartono N, Zeybek S, Pham DT. Using the Bees Algorithm to solve combinatorial optimisation problems for TSPLIB. IOP Conf. Series: Materials Science and Engineering 2020, 847: 012027.
[12] Koc E. Bees algorithm: theory, improvements and applications. Ph.D. thesis, Faculty of Engineering, Cardiff University, UK, 2010.
[13] Zeybek S, Ismail AH, Hartono N, Caterino M, Jiang K. An Improved Vantage Point Bees Algorithm to Solve Combinatorial Optimization Problems from TSPLIB. In Macromolecular Symposia 2021, 396(1): 2000299.
[14] Colak I, Bulbul HI, Sagiroglu S, Sahin M. Modeling a permanent magnet synchronous generator used in wind turbine and the realization of voltage control on the model with artificial neural networks. IEEE International Conference on Renewable Energy Research and Applications (ICRERA) 2012, 1-6.
[15] AMA Toolkit for LabVIEW - https://www.ni.com/en-gb/support/downloads/tools-network/download.ama-toolkit-for-labview.html#. 25.04.2022
[16] LabVIEW Optimization Vıs - https://www.ni.com/docs/en-US/bundle/labview-2021/page/gmath/optimization_vis.html#. 25.04.2022.
[17] Aria M. Educational simulator for teaching of particle swarm optimization in labview. TELEKONTRAN 2013, 1(1): 1-15.
[18] Thakur KS, Kumar V, Rana KPS, Mishra P, Kumar J, Nair SS. Development of bat algorithm toolkit in labview. International Conference on Computing, Communication and Automation (ICCCA) 2015, Greater Noida, India, pp. 5-10.
[19] Gupta S, Kumar V, Rana KPS, Mishra P, Kumar J. Development of ant lion optimizer toolkit in labview. 1st International Conference on Innovation and Challenges in Cyber Security (ICICCS) 2016, Greater Noida, India, 2016, pp. 251-256.
[20] Gupta S, Rana KPS, Kumar V, Mishra P, Kumar J, Nair SS. Development of a grey wolf optimizer toolkit in labview. 1st International conference on futuristic trend in computational analysis and knowledge management (ABLAZE) 2015, Greater Noida, India, 2015, pp. 107-113.
[21] Baronti L, Castellani M, Pham DT. An analysis of the search mechanisms of the bees algorithm. Swarm and Evolutionary Computation 2020, 59: 100746.
[22] Şahin M. Improvement of the Bees Algorithm for Solving the Traveling Salesman Problems. Bilişim Teknolojileri Dergisi 2022, 15(1), 65-74.
[23] MP-TESTDATA. The TSPLIB Symmetric Traveling Salesman Problem Instances. Retrieved from http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html . 10.01.2022.
[24] Castellani M, Otri S, Pham DT. Printed circuit board assembly time minimisation using a novel Bees Algorithm. Computers & Industrial Engineering 2019, 133: 186–194.
[25] Lambiase A, Iannone R, Miranda S, Lambiase A. Pham DT. Bees algorithm for effective supply chains configuration. International Journal of Engineering Business Management 2016, Volume 8: 1–9.
[26] Demiral MF. Analysis of a Hybrid Whale Optimization Algorithm for Traveling Salesman Problem. Mehmet Akif Ersoy Üniversitesi Fen Bilimleri Enstitüsü Dergisi 2021, 12(Ek (Suppl.) 1), 469-476.
[27] Şahin Y. Sezgisel ve Metasezgisel Yöntemlerin Gezgin Satıcı Problemi Çözüm Performanslarının Kıyaslanması. Bolu Abant İzzet Baysal Üniversitesi Sosyal Bilimler Enstitüsü Dergisi 2019, 19(4), 911-932.

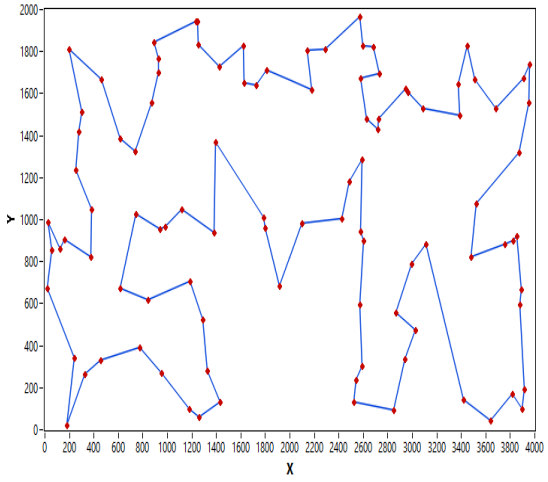
Ek – Bazı GSP'ler için hesaplanan rotalar



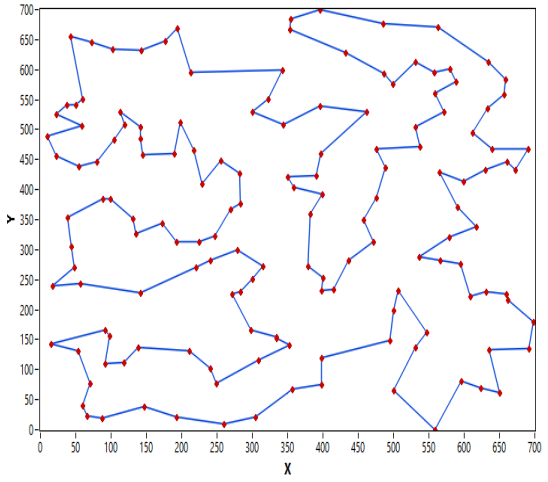
Berlin52



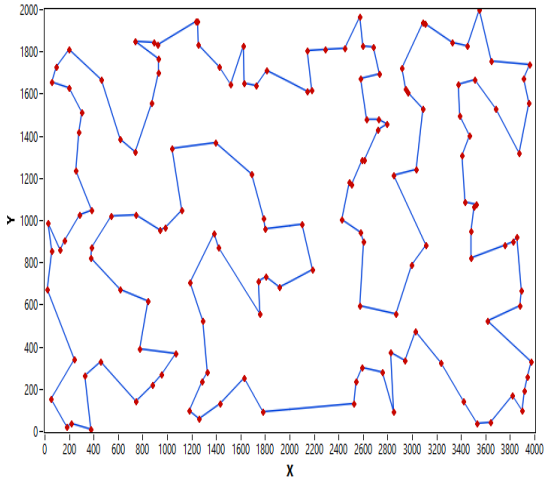
Pr76



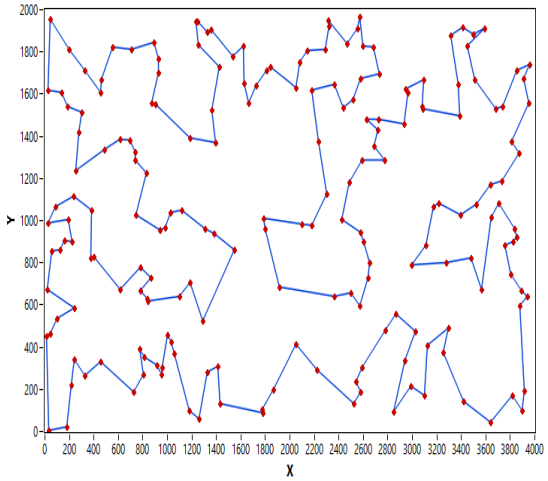
KroA100



Ch150



KroA150



KroA200

Şekil 11. Berlin52, Pr76, KroA100, Ch150, Kroa150, KroA200