Gezgin Satıcı Probleminin İkili Kodlanmış Genetik Algoritmalarla Çözümünde Yeni Bir Yaklaşım

Mehmet Ali Aytekin[1], Tahir Emre Kalaycı[2]

1 Sanko Holding, Bilgi İşlem Departmanı, Gaziantep , m.ali.aytekin@gmail.com
2 Ege Üniversitesi, Bilgisayar Mühendisliği Bölümü, İzmir, tahir.kalayci@ege.edu.tr

Özet: Gezgin Satıcı Problemi (GSP) kombinatöryel eniyileme ve global sezgisel arama alanlarında yoğun bir şekilde araştırılan ve çalışılan bir problemdir. Farklı türdeki GSP'ler için çok sayıda ve farklı çözüm yöntemleri vardır. Gezgin satıcı problemi alanında kullanılan önemli yöntemlerden biri de genetik algoritmalardır (GA). GSP sezgisel bir yöntem olan genetik algoritmaları karşılaştırmaya ve değerlendirmeye yardımcı olur. Bu çalışmada Gezgin Satıcı Problemi için bir genetik algoritma geliştirilmiş, geliştirilen yöntemin avantajları ve dezavantajları var olan yöntemler temel alınarak anlatılmıştır. Geliştirilmiş yöntemin en önemli yeniliği yeni bir ikili kodlanmış gen yaklaşımı kullanmasıdır. Yapılan deneysel çalışmaların bir kısmı bildiride sunularak sonuca ulaşılmıştır.

Anahtar Sözcükler: Genetik Algoritmalar, Gezgin Satıcı Problemi, Eniyileme, İkili kodlama

Solving Traveling Salesman Using Binary Encoded Genetic Algorithms

Abstract: Traveling Salesman Problem (TSP) is widely studied and researched problem in combinatorial optimization and global search heuristics. There are many and different solution techniques for different kind TSP's. An important technique for TSP is the genetic algorithms (GA). TSP helps to benchmark and evaluate GAs, which is a heuristics method. In this study, a new genetic algorithm technique has been developed to solve traveling salesman problem. Advantages and disadvantages of this technique has been explained by comparing with existing techniques. The most important novelty of this technique is using a new binary encoded chromosome approach. Conclusion is achieved by presenting some of the conducted experiments.

Keywords: Genetic Algorithms, Traveling Salesman Problem, Optimization, Binary encoding

1. Giriş

Gezgin satıcı problemi (GSP) verilen N düğüm (şehir) için, her düğüme bir kez uğramak şartıyla tekrar başlangıç düğümüne geri dönen en kısa (en az maliyetli) rotayı bulma problemidir[1]. Tanımlaması son derece kolay fakat çözümü NP-Zor bir problemdir [1]. Bir eniyileme problemi olan bu problemde bir çizge üzerine yerleştirilmiş noktalar ve aralarındaki maliyetler göz önüne alınarak her düğüme yalnız bir kere uğramak şartıyla en uygun maliyetle çizgedeki tüm düğümlerin dolaşılması olarak tanımlanabilir. Bu problemin çözümü bir Hamilton Döngüsü olarak da görülebilir [11].

GSP'yi matematiksel olarak iki şekilde tanımlayabiliriz: çizge problemi olarak ve permütasyon problemi olarak[2]. Eğer GSP' yi çizge problemi olarak ifade edersek; G=(V,E) çizgesi ve bu çizgede tanımlanmış tüm Hamilton döngüleri F ile temsil edilmiş olsun. Her e E için daha önceden belirtilmiş bir ağırlık () değeri vardır. İşte GSP bu G çizelgesinde en kısa maliyetle, tüm düğümlere uğrayarak, bir tur (Hamilton döngüsü) elde etmeyi amaçlayan bir problemdir[2].

Gezgin satıcı problemi (GSP) yapay zeka ve eniyileme alanında en çok araştırılan ve çözümler üretilen, algoritmalar geliştirilen bir problemdir. Teorik bilgisayar bilimleri ve işletimsel araştırmada ("operational research") kombinasyonel eniyileme problemidir. Çok farklı eniyileme yöntemleri, yapay zeka teknikleri bu probleme yönelik olarak geliştirilmiştir. Gezgin satıcı problemi alanında kullanılan önemli yöntemlerden biri de genetik algoritmalardır (GA). Sezgisel bir yöntem olan genetik algoritmalar açısından GSP bir karşılaştırma ve değerlendirme ölçütü işlevi de görmektedir. Ayrıca sadece GA'ların değil, diğer tüm yeni algoritmaların (karınca sistemi, evrimsel yöntemler, sinir ağları, tabu arama, benzetimli tavlama, açgözlü aramalar, vb.) karşılaştırılması, değerlendirilmesi ve ölçülmesi için de kullanılmaktadır [22,23]. Bu sezgisel yöntemler makul bir sürede iyi sonuçlar sağlamaktadırlar. Johnson vd. [24] yaptıkları çalışmada yaklaşık sonuçlar üreten bazı yerel arama metotlarını da incelemişlerdir. GSP'yi çözmek için önerilen algoritmalar iki başlık altında toplanabilir[1]: Kesin Algoritmalar ("Exact algorithms") ve Sezgisel Algoritmalar ("Heuristics algorithms").

Kesin Algoritmalar: Bu algoritmalar, genellikle, GSP'nin tamsayılı lineer programlama formülünden türetilen yaklaşımlardır. Fakat bu algoritmalar hesaplanabilirlik açısından pahalıdırlar[1]. Bu yaklaşıma örnek olarak "Branch & Bound" algoritmasını örnek olarak verilebilir. İlk akla gelen yöntem her olasılığın denendiği yöntemdir. Diğer yöntemler olarak lineer programlama yöntemleri, dinamik programlama yöntemleri sayılabilir [21][1] .

Sezgisel Algoritmalar: Kesin algoritmaların çalışması verimli olmayabilir. Bu durumda, ideal çözüme yakın bir çözümü, kesin algoritmaları kullanmadan da bulabiliriz. Pratikte sezgisel algoritmalar, kesin algoritmalara tercih edilmektedir[1]. GSP' yi çözen sezgisel algoritmaları üçe ayırabiliriz: Tur oluşturan sezgiseller, Turu geliştiren sezgiseller ve bu iki yöntemin melez şekilde kullanıldığı sezgiseller[1].

- **Tur oluşturan Sezgiseller:** Tur oluşturan algoritmaların ortak özelliği, bir sonuç buldukları zaman, bu sonucu geliştirmek için uğraşmamalarıdır[3]. Bu noktada algoritmaların çalışması sona erer. Bilinen tur oluşturan sezgisel algoritmalar şunlardır: En yakın komşu, Greedy, Ekleme Sezgiseli ve Christofides algoritmalarıdır[3]. Bu algoritmaların en optimum değeri %10-15 arasıdır[3].

- **Turu geliştiren Sezgiseller:** Bu algoritmalar turu geliştirmeyi amaçlarlar. Bu algoritmalara örnek olarak 2-opt, 3-opt ve Lin-Kernighan gibi yerel eniyileme ("optimization") algoritmalarını örnek verebileceğimiz gibi, Tabu araması, Genetik algoritmalar, Benzetim Tavlama ve Karınca Kolonisi Algoritması gibi Yapay Zeka yöntemlerini de örnek olarak verebiliriz.

- **Melez Yöntemler:** Hem tur oluşturma hem de tur geliştirme sezgisellerini bir arada kullanan algoritmalardır. Bunlara örnek olarak "Yinelemeli ("iterated") Lin-Kernighan[10]"'ı örnek verebiliriz. En başarılı sonuçlar melez yöntemlerden elde edilmektedir[1].

GSP'yi çözmek üzere en sık kullanılan evrimsel hesaplama yöntemi genetik algoritmalardır. Bir çok NP-tam problem için GA'nın oldukça başarılı meta-sezgisel bir yöntem olduğu ispatlanmıştır [22].

Bu çalışmada GSP için önerilen çözüm yöntemlerden biri olan Genetik Algoritmalar detaylı olarak ele alınacaktır. Daha sonra ise GSP' yi çözmek için geliştirdiğimiz ve yeni bir yöntem olan İkili Kodlu Genetik Algoritmalar tanıtılacak ve diğer yöntemlerle karşılaştırılıp, yapılan deneysel çalışmalar ortaya konacaktır.

2. Genetik Algoritmalar

Genetik Algoritmalar (GA), arama ve eniyileme problemlerini çözmekte kullanılan uyarlanabilir bir yöntemdir[5]. Doğada bireyler yiyecek, su ve barınak gibi kaynaklar için yarış halindedirler. Aynı zamanda her birey soyunu devam ettirmek ister. İşte bu şartlarda, yarışı kazanan (güçlü ve çevreye en iyi uyumu gösteren bireyler) hem kaynaklara sahip olur hem de soyunu devam ettirme şansını elde eder. Yarışı kazanan bireylerin ürettiği yeni bireyler de, atalarından gelen özellikleri de alırlar [5].

GA, yukarıda bahsettiğimiz "doğal seçilim ve uyarlanım" prensibinden esinlenilerek ortaya atılmış bir yaklaşımdır. Bireyler, GA'da ilgili problemin çözümlerini temsil etmektedir. Bireylerin (yani kromozomların) ortama uyum sağlama ve hayatta kalma durumu (yani bireyin "uygunluk değeri") ise, GA'da, ilgili çözümün problemi çözebilme yeteneğini temsil etmektedir. GA mevcut birçok çözümden en uygunlarını seçerek, bu çözümlerden yeni çözümler elde etmeyi amaçlar (Şekil 1). Algoritmanın sonlanması (durma-koşulu) ise şu üç durumdan birine bağlı olarak gerçekleşir: belli bir döngü sayısına ulaşma, en iyi sonucun hiç değişmemesi veya neslin ortalama uygunluk değerinde herhangi bir değişikliğin gözlemlenmemesidir[7].

GA'nın en önemli avantajı paralel çalışmasıdır. Büyük problemler için bu önemli bir kazançtır. Çok geniş bir çözüm uzayını hızlı bir şekilde arayabilir. Geniş bir problem aralığında kullanılabilirler. Başlangıçta problem hakkında bilgi olmadan da problemleri çözebilir. GA'nın gerçekten iyi çalışması için olası çözümlerin iyi temsil edilmesi, uygunluk fonksiyonunun, işleçlerin ve parametrelerin iyi belirlenmesi başarım için şarttır. GA'da yerel en iyi değer yakınsamanın bile garantisi yoktur. GA'da oluştuğu bilinen en önemli problem erken yakınsamadır. Bunun anlamı diğer bireylerden çok daha iyi olan bir birey tüm çözüm arama sürecini etkisi altına alarak, gerçek çözüme (global optima) yaklaşılmasını engelleyebilir, yerel en iyi değerde (local optima) takılıp kalınmasına neden olur [17,18, 19]. GA'nın dezavantajlarını aşmak için yerel arama yöntemleri ve diğer eniyileme yöntemleriyle melez yöntemler geliştirilmektedir.
GA'nın basit algoritması Şekil 1'de görülebilir. GA gerçekleştirimleri bu basit taslağın benzerlerini, değiştirilmiş sürümlerini kullanırlar.

---

1. Konuyla ilgili olarak http://en.wikipedia.org/wiki/Travelling_salesman_problem#Exact_algorithms adresinden daha fazla bilgi edinilebilir.

```
BEGIN GA
  t:=0
  populasyon P(t)' yi ilkle
  P(t)'deki her bireyin uygunluk değerini hesapla
  WHILE (NOT durma-koşulu) DO
    BEGIN WHILE
    t :=t+1
    P(t-1) den P(t) 'yi oluştur
    P(t)'ye GA uygula.
    P(t)'deki her bireyin uygunluk değerini hesapla
  END WHILE
END GA
```

Şekil *1* - *Genetik Algoritma Sözdekodu [8]*

Genetik Algoritmaları uygulama aşamasında aşağıdaki adımlara karar verilmesi son derece önemlidir[5,7,9]:

- Kodlama
- Çözümleri Değerlendirme (Uygunluk Fonksiyonu)
- Birey Üretimi (Çaprazlama ve Mutasyon, Seçilim ve üretilen bireylerin popülâsyona eklenmesi)

2.1. Kodlama

Çözümdeki parametrelerin ("gen"lerin) nasıl temsil edileceğidir. Bu parametreler, yani genler ikili temsil, tamsayılar, kayan noktalı sayılar, ağaç veri yapısı, dizi vs. olarak temsil edilebilmektedir. Bu temsil işlemi probleme bağlı olan bir işlemdir.

2.2. Çözümleri Değerlendirme

GA'daki her çözümün yani kromozomun, problemi ne derecede çözebildiğini hesaplamamızı sağlayan fonksiyona "uygunluk fonksiyonu" denir. Uygunluk fonksiyonu problem bağımlı bir fonksiyondur ve GA'nın en önemli kavramlarından biridir.

2.3. Birey Üretimi

Yeni bireylerin üretimi GA'nın kalbidir. Yeni üretilen bireylerin, ata bireylere göre daha yüksek uygunlukta olması beklenir. Birey üretiminde üç önemli karar aşaması vardır: ata bireyleri seçme(seçilim), yeni bireyleri üretme(çaprazlama ve mutasyon) ve üretilen yeni bireyleri populasyona ekleme işlemi[9].

2.3.1. Seçilim

Seçilim işlemi, yeni bir birey üretmek için popülasyondan var olan bireyleri seçme işlemidir. Seçilim metodunun nasıl olduğu, GA' ın yakınsamasını da belirlemektedir[9]. Bazı seçilim yöntemleri Sıralama Seçilim, Rastgele Seçilim, Rulet Tekerleği ve Boltzmann Seçilimidir.

2.3.2. Yeni birey Üretimi

Ata bireyler seçildikten sonra, önemli bir aşama da, seçilen bu bireylerden yeni bireylerin nasıl üretileceğidir. Bu aşama çaprazlama olarak da bilinir. Çaprazlama işleminde daha önce incelenmemiş olan çözümlerin incelenmemesi ve yüksek uygunlukta çözümlerin üretilmesi beklenir. Üretilen bir çözümde, belli genlerin değerleriyle oynama işlemi de "mutasyon" olarak algılanmaktadır. Mutasyon işleminin amacı, GA'yı yerel minimum değerlerinden kurtarmaktır.

2.3.3. Üretilen bireyleri populasyona ekleme

Bu aşamada karar verilmesi gereken şey, üretilen yeni bireylerin, yani çocukların, populasyona nasıl ekleneceğidir. Bazı yaklaşımlar şunlardır: populasyondaki kötü bireyler ile çocuk bireyler yer değiştirebilir, çocuk bireyler ile ata bireyler yer değiştirebilir, veya çocuk bireyler ile önceki populasyon birleştirilip, bu son populasyondan ilk "n" birey yeni populasyon olarak alınır. Ayrıca her yeni nesil üretiminde, bir önceki populasyonun en iyi bireylerinin elde tutulup tutulmayacağı (seçkincilik - "elitism") da karar verilmesi gereken önemli bir durumdur.

3. İkili Kodlu Genetik Algoritma

Bu çalışmada geliştirdiğimiz yöntem İkili Kodlu Genetik Algoritmalar'dır. Tabii bu yöntem Lidd'in geliştirmiş olduğu İkili Kodlu Genetik Algoritmalar'dan farklıdır. 4.1'de Lidd'in geliştirdiği, 4.2' de ise bizim geliştirdiğimiz yöntem tanıtılacak. 4.3. de geliştirilen yöntem, GSP' yi çözmek için kullanılan diğer GA yöntemleri karşılaştırılacak; 4.4'de de geliştirilen yöntem avantaj ve dezavantajları ortaya konulacaktır.

3.1. Lidd' in Geliştirdiği İkili Kodlu Genetik Algoritma

n,şehirli GSP' nin ikili temsilinde, her şehri diğerlerinden farklı olacak şekilde temsil etmek için 'lik bit dizisine ihtiyacımız olacaktır. Böylece kromozom boyutu da n* olacaktır. Örnek vermek gerekirse 6 şehirlik bir GSP, ikili temsille aşağıdaki gibi kodlanır.

i	Şehir i	i	Şehir i
1	000	4	011
2	001	5	100
3	010	6	101

Tablo 1 - *Şehirler ve İkili Temsil Olarak Karşılıkları*

Burada dikkat edilmesi gerekilen nokta, hem her şehri ayrı ayrı ikili değer ile temsil etmemiz hem de 110 ve 111 3-bitlik string'ler olmasına rağmen gen olarak tanımlanmamış olmasıdır. Örneklerde kullanacağımız kromozomlar (000 001 010 011 100 101) ve (101 100 011 010 001 000) olsun.

3.1.1. Klasik çaprazlama

John Holland'ın tanımladığı [20] klasik çaprazlamada iki kromozom üzerinden rasgele bir çaprazlama noktası seçilir ve her kromozom bu çaprazlama noktasından alt parçalara ayrılır. Daha sonra kromozomlar arasında farklı parçalar yeniden birleştirilir. Farzedelim ki 9. ve 10. bitler arasını çaprazlama noktası olarak seçilsin:
(000 001 010 | 011 100 101)
(101 100 011 | 010 001 000)

Şimdi kromozomlaradaki farklı parçaları birleştirip çocuk kromozomları üretelim. (000 001 010 010 001 000) ve (101 100 011 011 100 101) yeni bireylerimiz olacaktır. Fakat bu bireyler geçerli bir tur oluşturmuyor. Bu durumda bunları düzeltmek için bir "tamir edici algoritma" (repair algorithm) ya ihtiyaç duyarız[1].

3.1.2. Klasik Mutasyon

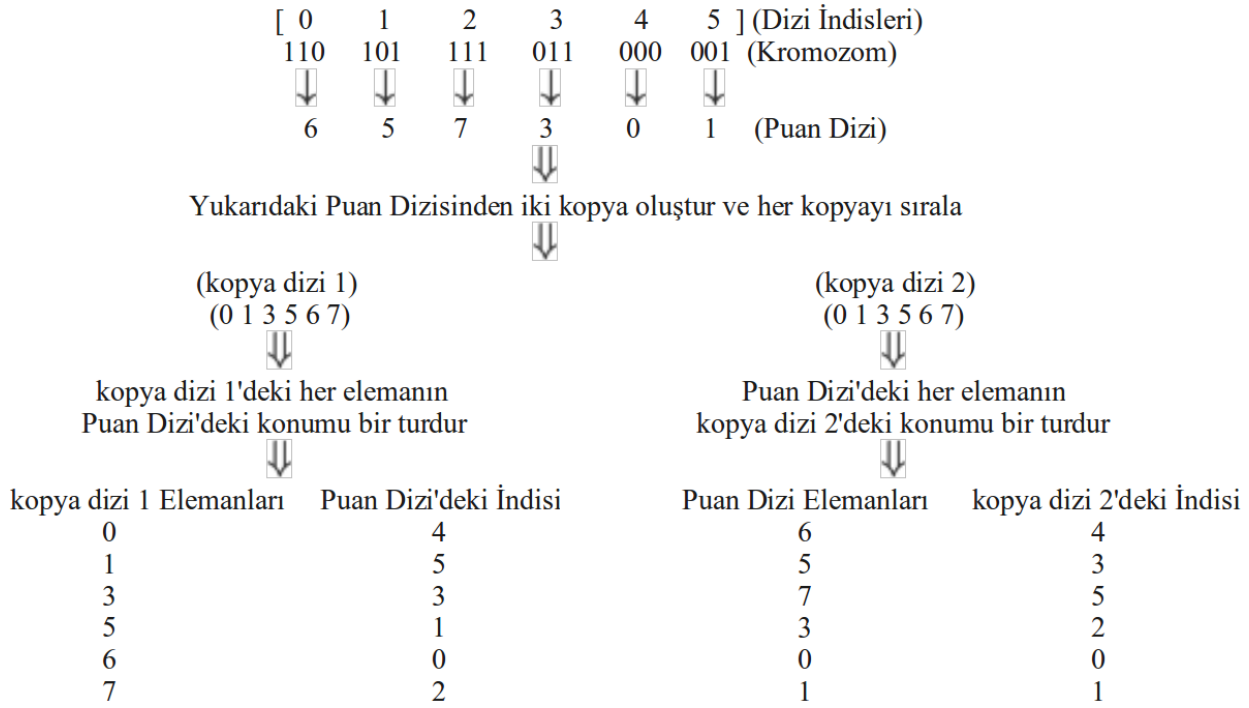
Yine John Holland'ın [20] tanımladığı klasik mutasyon operatörü belli bir olasılıkla (mutasyon olasılığı), bir veya bir kaç genin değerini değiştirir. Mesela şu kromozom üzerinde mutasyon işlemi uygulayalım:
(000 001 010 011 100 101)
Diyelimki birinci ve ikinci bit mutasyon işlemi için seçilmiş olsun. Bu durumda bu bitlerdeki değerler 0'dan 1'e dönüşür. Böylece mutasyona uğramış kromozomumuz (110 001 010 011 100 101) halini alır ki bu da geçerli bir tur değildir. Düzeltilmesi gerekir
Lidd'in çalışması küçük boyutlu veriler (100 şehirli GSP' ye kadar) için iyi sonuçlar üretse de, Whitley'ın da ifade ettiği gibi GSP için önerilen bir temsil şekli olamamaktadır[16]. Whitley GSP'nin ikili temsil ile sıra bağımlılığından kurtararak veya işleçlerin anlamlı bir şekilde uygulanabildiği şekilde temsil etmenin pratik bir yolu olmadığını söylemektedir Bu nedenle uygulanan işleçlerin hatalı kromozomların üretilmesine neden olduğunu, çaprazlama ve mutasyon sonucunda tekrarlı ya da eksik şehirler içeren çözümler üretildiğini belirtmektedir. Sonuç olarak bu problemi çözmek için, standart genetik çaprazlama işleçlerinin değiştirilerek (bir bakıma probleme uyarlanarak)

kullanılması gerektiğini savunmaktadır. İdeal üretim tekniklerinin, atalardan bilgiyi bozmadan tutarlı bir şekilde alması gerektiğini çalışmasında [16] yazmıştır.

3.2. Geliştirilen Yeni Yöntem

Biz geliştirdiğimiz yöntemde, Lidd'in yaptığını aksine her şehre ikili temsilde kodlanmış bir permütasyon atamanın aksine, bir puan değeri atıyoruz. Böylece her genin, kromozom içinde elde ettiği bir puan değeri olmuş oluyor. Daha sonra bu puan değerleri bizim belirlediğimiz kriterlere göre sıralanarak, geçerli bir tur elde ediyoruz. Burada önem çekmek istediğimiz olay, her gene ait bir puan değerinin olmaması (yani permütasyon oluşturma çabası yok), hatalı ya da eksik gen üretimi gibi olayların ortadan kalkmış olmasıdır.

6 şehirlik bir GSP düşünelim. Bu problemde her şehri temsil etmek için kullanmamız gereken bit sayısı $log_2 6$' dır. Bu değeri en yakın büyük tam sayıya yuvarlarsak 3 bit bizim için yeterli olacaktır. O halde kromozom boyutumuz da 18 (3*6) olacaktır. Rasgele bir kromozom üretelim: (110 101 111 011 000 001). Simdi bu kromozomdan turu nasıl elde edeceğiz? Bunun için iki yöntem vardır. Ama her iki yöntemden önce her bir genin onluk sistemdeki puanını hesaplamak gerekecektir(bunu puan_dizi olarak adlandıralım). Genlerin puan durumu şu şekildedir: (6 5 7 3 0 1).
Ve son olarak puan dizisinin bir kopyasını(bunu da kopya_dizi olarak adlandıralım) elde edelim ve bu kopyayı küçükten büyüğe doğru sıralayalım: (0 1 3 5 6 7).
Yöntem 1: Kopya dizideki her bir elemanı referans alarak, bu elemanın puan dizideki konumu tur için bir şehri temsil edecektir. Bu durumda kopya dizideki 0 elemanının puan dizideki konumu 4, kopya_dizideki 1 elemanının puan dizideki konumu 5, ... olarak hesaplarsak elde ettiğimiz tur:4-5-3-1-0-2 olacaktır.
Yöntem 2: Puan dizideki her bir elemanı referans alarak, bu elemanın kopya dizideki konumu tur için bir şehri temsil edecektir. Bu durumda puan dizideki 6 elemanının kopya dizideki konumu 4, puan dizideki 5 elemanının kopya dizideki konumu 3, ... olarak hesaplarsak elde ettiğimiz tur:4-3-5-2-0-1 olacaktır.
Yukarıda detaylarını açıkladığımız her iki tur üretme yönteminde de ne hatalı ne de eksik bir şehir üretiyoruz. Böylece ikili temsil için tanımlı olan çaprazlama ve mutasyon işleçlerini, herhangi bir tamir edici algoritmaya gereksinim duymadan kullanabiliyoruz. Yöntemleri Şekil 2'deki görselleştirmeden inceleyebilirsiniz.



Şekil 2 *- Geliştirdiğimiz yöntemin adımları*

3.2.1. Genetik Algoritma

Genetik Algoritma'da çaprazlama yöntemi olarak "Uniform Çaprazlama" tekniğini kullandık. Yani çaprazlanacak iki bireyin herbiri, çocuk bireye genini aktarmak için .5 şansa sahiptir. Mutasyon işleminde ise bir kromozomun her bir geni .5 olasılıkla mutasyona uğrayabilir. Seçilim işleminde ise çocuk bireyler ile daha önceki populasyonun birleştiriliyor ve bu birleşim sıralanarak populasyon boyutu kadar birey alınıp yeni populasyon olarak genetik algoritmaya dahil edilmektedir. Tabi bu işlem sırasında "elitisim" uygulayarak, en iyi 3 bireyi her zaman muhafaza ediyoruz.

3.3. Diğer Temsil Yöntemleriyle Karşılaştırılması

N şehir sayısı olmak üzere aşağıdaki tabloda yöntemleri karşılaştırabiliriz.

TEMSİL YÖNTEMİ	GEN NE ANLAMA GELİR?	KROMOZOM NE ANLAMA GELİR?	HATALI KROMOZOM ÜRETİMİ?	KROMOZOM BOYUTU
Permütasyon Kodlama	Ziyaret edilecek şehir	Tur	Evet	N
Matris Kodlama	İki şehrin komşu olup olmaması	Tur	Evet	N*N
İkili Temsil (Lidd's way)	Ziyaret edilecek şehir	Tur	Evet	N*$log_2 N$
İkili Temsil (geliştirilen yöntem)	Gen indisindeki şehrin sıralamadaki değeri	Her bir şehir için hesaplanan puan değeri	Hayır	N*$log_2 N$
Reel Kodlama	Gen indisindeki şehrin sıralamadaki değeri	Her bir şehir için hesaplanan puan değeri	Hayır	N

Tablo 2 *- Yöntemlerin karşılaştırılması*

3.4. Avantaj ve Dezavantajları

Bu yöntemin **avantajlarını** aşağıdaki gibi sıralayabiliriz:

- İkili temsil ile gerçekleştirmemiz sayesinde problem "permütasyon bulma problemi" olmaktan öte, bir "değer bulma problemine" dönüşmüştür.
- Klasik GA kullanarak problemi çözüyoruz. Böylece özel çaprazlama ya da mutasyon operatörleri geliştirme zorunluğu ortadan kalkmış oluyor.
- Hatalı kromozom üretilmesi olayı ortadan kalkmaktadır (gerek permütasyon kodlamada gerekse de Lidd'in yönteminde çaprazlama sonucu hatalı kromozomlar üretilmekteydi.).
- Lidd'in geliştirdiği ikili temsilde, gen olarak tanımlanmamış string ifadeler üretilebilmekteydi (mutasyon aracılığıyla). Bizim geliştirdiğimiz yöntemde ise böyle bir şey söz konusu değil.

Dezavantajları ise

- Şehir sayısına bağlı olarak kromozom boyutu üssel bir şekilde artmaktadır. Bu da işlem süresini uzatmaktadır.

4. Deneyler

Geliştirdiğimiz yöntemi test etmek için TSPLIB[2]' ten gr24, gr48, berlin52 ve kroA 100 test verilerini kullandık. Her test verisi için algoritmayı, Yöntem-1'e göre çözümleyip 10 defa çalıştırdık ve aşağıdaki sonuçları elde ettik.

---

2. http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/ adresinden indirilebilir.

TSPLIB Problemi	POPULASYON BOYUTU	NESİL SAYISI	HATA YÜZDESİ(yaklaşık olarak)
gr24	500	300	% 8
gr48	2000	400	% 11
berlin52	6000	600	% 13
kroA100	10000	800	% 25

5. Sonuç

Geliştirdiğimiz yöntemin kendine özgü avantajları ve dezavantajları olduğu ortadadır. Yöntem üzerinde bazı soruların sorulması ve yanıtlanması sonucu avantajlar arttırılarak, dezavantajların azaltılması sağlanabilir. Bu nedenle gelecekte aşağıdaki sorulara yanıtlar aranması, yöntemin iyileştirilmesi yönünde çalışmalar yapılması planlanmaktadır:

- Daha az ikili değer kullanılarak bu problem çözülebilir mi?
- Artan şehir sayılarına göre, ikili temsilin kendine özgü diğer çaprazlama ve mutasyon işlemlerinde nasıl bir sonuç elde edilir? Dikkat edilirse şehir sayısı artıkça, algoritma verimliliği azalmaktadır. Bu problemi de çözmemiz gerekecektir.
- Bu yönteme uygun yerel eniyileme algoritmaları geliştirebilir miyiz?
- Diğer temsil yöntemlerine göre nasıl çalışmaktadır?

6. Kaynaklar

[1] J.-Y. Potvin, Genetic algorithms for the travelling salesman problem, forthcoming in Annals of Operations Research on "Metaheuristics in Combinatorial Optimization", eds. G.Laporte and I.H. Osman (1996).
[2] G. Gutin, A.P. Punnen (Eds.), The Travelling Salesman Problem and its Variations, Kluwer Academic Publishers, Dordrecht, 2002.
[3] Christian Nilsson, Heuristic Algorithms For Travelling Salesman Problem, Linköping University. Son Erişim: 09.12.2009 Erişim bağlantısı: http://www.ida.liu.se/~TDDB19/reports_2003/htsp.pdf
[4] Larrañaga, P., Kuijpers, C. M., Murga, R. H., Inza, I., and Dizdarevic, S. 1999. Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators. *Artif. Intell. Rev.* 13, 2 (Apr. 1999), 129-170.
[5] Beasley D., Bull, D.R., Martin, R.R., 1993a. An Overview of Genetic Algorithms: Part 1, Fundamentals. University Computing, Vol.15(2), pp. 58-69, UK.
[6] Beasley D., Bull, D.R., Martin, R.R., 1993a. An Overview of Genetic Algorithms: Part 2, Research Topics. University Computing, Vol. 15(4), pp. 170-181, UK.
[7] Kylie Bryant, Genetic Algorithms and Travelling Salesman Problem, Senior Thesis, Dept. Of Mathematics, Harvey Mudd College, 2000. Son Erişim: 09.12.2009 Erişim bağlantısı: http://www.math.hmc.edu/seniortheses/01/bryant/finalthesis.pdf
[8] Michalewicz, Z. 1996 *Genetic Algorithms + Data Structures = Evolution Programs (3rd Ed.)*. Springer-Verlag.
[9] Sivanandam, S. N. and Deepa, S. N. *Introduction to Genetic Algorithms*. 1st. Springer Publishing Company. 2007
[10] Johnson, D. S. 1990. Local Optimization and the Traveling Salesman Problem. In *Proceedings of the 17th international Colloquium on Automata, Languages and Programming* (July 16 - 20, 1990). M. Paterson, Ed. Lecture Notes In Computer Science, vol. 443. Springer-Verlag, London, 446-461.
[11] Kalaycı, T.E. *Yapay Zeka Teknikleri Kullanan Üç Boyutlu Grafik Yazılımları İçin "Extensible 3D" (X3D) İle Bir Altyapı Oluşturulması ve Gerçekleştirimi*, Ege Üniversitesi Bilgisayar Mühendisliği Yüksek Lisans Tezi, 2006.
[12] Obitko, Marek and Slavík, Pavel., 1999, Visualization of Genetic Algorithms in a Learning Environment, Spring Conference on Computer Graphics, SCCG'99, p. 101-106.
[13] Dorigo, M., Gambardella, L.M., 1997, Ant colony system: a cooperative learning approach to the traveling salesman problem, Evolutionary Computation, IEEE Transactions on, 1(1):53-66
[14] Lutton, J.L., Bonomi, E., 1984, The N-City Travelling Salesman Problem: Statistical Mechanics and the Metropolis Algorithm, SIAM Review, 26:551-568
[15] Fogel, D.B., 1988, An Evolutionary Approach to the Traveling Salesman Problem, Biological Cybernetics, 60:139-144
[16] Whitley, D., Starkweather, T., and Fuquay, D. 1989. Scheduling problems and traveling salesman: the genetic edge recombination. In Proceedings of the Third international Conference on Genetic Algorithms (George Mason University, United States). J. D. Schaffer, Ed. Morgan Kaufmann Publishers, San Francisco, CA, 133-140.

[17] Alexandre Weffort Thenorio, Genetic Algorithms, Son Erişim: 04.12.2009, Erişim bağlantısı: http://www.cs.chalmers.se/Cs/Grundutb/Kurser/algsem/Projects2007/GeneticAlgorithms.pdf
[18] Genetic Algorithms, Son Erişim: 04.12.2009, Erişim bağlantısı: http://www.tjhsst.edu/~ai/AI2001/GA.HTM
[19] Max Moorkamp, Genetic Algorithms A Step by Step Tutorial, Kasım 2005, Barcelona, Son Erişim: 04.12.2009, Erişim bağlantısı: http://www.dias.ie/~mm/ga_tutorial.pdf
[20] Holland, J. H. Adaptation in Natural and Artificial Systems. University of Michigan Press: Ann Arbor, MI. 1975.
[21] Applegate, D. L.; Bixby, R. E.; Chvátal, V.; Cook, W. J. (2006), The Traveling Salesman Problem: A Computational Study, Princeton University Press, ISBN 978-0-691-12993-8
[22] Uğur, A. Path planning on a cuboid using genetic algorithms. *Inf. Sci.* 178, 16 (Aug. 2008), 3275-3287. 2008.
[23] C.-F. Tsai, C.-W. Tsai, C.-C. Tseng, A new hybrid heuristic approach for solving large traveling salesman problem, Information Sciences 166 (1–4) (2004) 67–81.
[24] D.S. Johnson, L.A. McGeoch, The traveling salesman problem: a case study in local optimization, in: E.H.L. Aarts, J.K. Lenstra (Eds.), Local Search in Combinatorial Optimization, John Wiley & Sons, New York, 1997, pp. 215–310.