EVRİMSEL BİLGİ İŞLEME

Göktürk Üçoluk

Evrimsel Bilgi İşleme veya alternatif adlandırmaları ile Evrimsel Berim[1] (EB), Evrimsel Algoritma canlılar doğasında var olduğunu gözlemlediğimiz evrim mekanizmasının, bilgisayarlarda problem çözmede temel model olarak kullanıldığı işleme sistemlerinin adlandırılmasında kullanılan genel terimdir.

Farklı EB uygulamalarındaki ortak özellik herhangi bir problemin çözümünün bir veri yapısı ile betimlenmesi ve bu veri yapısı içindeki değerlerin doğal evrimdekine benzer yöntemlerle belirlenmesidir. Biyoloji biliminden esinlenilerek problemin çözümünü kodlayan veri yapısına *kromozom* veya *fenotip* denmektedir. Bir problem (çoğunlukla) birden fazla parametrenin değerinin belirlenmesi ile çözüm bulmaktadır. Bu parametrelerin her biri kromozomda yer alacaktır. Kromozomun öğesi olan ve herbiri problemin bir parametresini kodlayan birimlere *gen*; her bir genin alabileceği değerlerin kümesine o genin *aleli* denmektedir. EB de herbiri probleme farklı bir çözüm adayı olan kromozomlardan bir havuz tutulmakta ve bu havuz evrimsel yöntemlerle değişikliğe uğratılmaktadır. Kromozom biçemindeki ve problemin bir aday çözümü olan bu havuz öğelerine *birey* veya *genotip* denmektedir. Bir EB uygulamasının genel yapısı şöyledir:

- Birey havuzunu rasgelelikten de yararlanarak oluşturulması.
- Bir durma koşulu sağlanıncaya kadar aşağıdaki eylem dizisini yinelenmesi:
  - Havuzda oluşan bireylerin herbirisinin kodladığı problem çözümünün kalitesini değerlendirilmesi.
  - Bu değerlendirmenin ışığında evrimsel işlem yapıcılar (operatörler) uygulayarak havuzu değişikliğe uğratılması.

Evrimsel işlem yapıcıların sık kullanılanlarını özetlersek:

Seçim *(selection)* **:** Değişikliğe uğramış bir havuzdan bir sonraki kuşağa hangi bireylerin aktarılacağının seçimidir.

[1] 'Bermek' kökü ODTÜ Bilgisayar Müh. öğretim üyelerinin bazılarınca İngilizce 'Compute' kökünün Bilgisayar Bilimleri bağlamındaki anlamına karşılık olarak kullanılmaktadır. Orta Asya Türkçesinde anlamı olan bu kök, güncel Türkçemizde kullanılmamaktadır ve çağrışımsal bir anlam yükü de barındırmamaktadır. *Compute* sözcüğü bilgisayarın eylemini tanımlamaktan aslında uzak olduğundan yeni bir sözcük atama noktasında, başka (kısıtlayıcı) anlamlarla yaralı, İngilizceden yakın çeviri bir sözcüğün kullanımını uygun bulmamaktayız.

Üreme *(reproduction)* **:** Havuzdaki bireylerden bazılarının eşleştirilmesi ve bu *ata* olarak adlandırılan bireylerin gen bilgilerini içeren, *çocuk* olarak adlandırılan, yeni kromozomların üretilmesi. Böylece her çocuğun her bir geni atalarından birisi ile aynı olacaktır.

Değişikliğe uğratma *(mutation)* **:** Doğadakine pek benzer olarak, havuz bireylerinin arasından rasgele seçilenlerinin bazı gen değerlerinin rasgele değiştirilmesidir.

EB uygulamaları kromozomlarında kullandıkları veri yapılarına ve evrimsel yöntemlere göre farklılaşmakta ve çeşitli adlandırmalarla anılmaktadırlar:

Genetik Algoritma (GA) Kromozom için seçilen veri yapısı doğrusaldır. Problemin çözümü 0/1 lerden oluşmuş bir dizgidir *(string)*. Şüphesiz bu dizginin gen anlamındaki alt bölgeleri sayısal veya etiketsel özelliklerde olabilecektir. Kodlanmış parametrelerin eş türden olmak zorunlulukları yoktur. Örneğin iki tam sayı geni, üç noktalı sayı geni, bir renk kodlaması geni bir GA kromozomunu oluşturuyor olabilir.

Genetik Programlama (GP) Bu adlandırma tarihsel nedenlerle kapsamı tam olarak yansıtmamaktadır: kapsamın programlama ile ilgisi yoktur. GP, kromozom için seçilen veri yapısının özyinelemeli dinamik veri yapılarından (çoğunlukla da dinamik ağaç yapıları) olduğu uygulamaların genel adıdır. Gen kavramı alt veri yapısı (çoğunlukla alt ağaç) anlamındadır. Örneğin iki atalı en basit üremede ata kromozomlar yani ağaçlar rasgele birer yerlerinden budanır ve elde edilen iki alt ağaç değiş tokuş edilir (ve böylece iki yeni kromozom (ağaç), yani çocuk imal edilmiş olur)

Evrim Stratejisi (ES) Üremenin eşleştirmeye dayalı olarak değil de denetimli bir değişikliğe uğratmaya dayalı olarak yapıldığı, doğrusal veri yapılı, genlerin yalnızca sayısal değerlerden oluştuğu bir yöntemdir (Buradaki sayısal kavramı genin değerini değiştirmek bakımından aritmetik işlemlere tabi tutulabilmesi anlamındadır). Üremede her bireyin her bir gen değeri yine evrimsel yöntemle değiştirilen ve o bireyin o genine özgü olan bir standart sapmanın belirlediği bir normal olasılık dağılımına göre farklılığa uğratılır.

Kendini Uyarlayan Etmenler (KUE) Yapay yaşam uygulamalarının evrimsel yöntem kullananlarının genel adıdır. Burada problem nicelleştirilmiş yapay bir gereksinim, amaç ve tehlike ortamındaki yazılım etmenlerinin (agent) hayatta kalmaları ve ereklerini gerçekleştirmeleridir. Yazılım etmenlerinin nasıl kodlandıklarının önemi bulunmamaktadır. Önemli olan geliştirdikleri davranış biçimlerini evrimsel yollarla sonraki kuşaklara aktarmalarıdır.

Tüm EB uygumalarında ortak olan bir kavram da *uygunluk değerlendirmesidir (fitness evaluation)*. Bütün çözüm bulma yöntemleri gibi EB de, her bir gen için tanımlı değer kümelerinin kartezyen çarpımından oluşan arama uzayında gezinmektedir. Şüphesiz bu gezinmenin rasgele olmaması için çözümün ne olduğu, daha doğrusu neyin iyi neyin kötü çözüm olduğunun tamamen nicel bir tanımının yapılmasına gereksinim vardır. Bu

tanım EB'nin özellikle seçim ve üreme işlemlerinde kullanılmak durumundadır. Örneğin havuzun en iyilerine bir kuşak daha yaşam hakkı tanımak, iyilere (kötülere göre) daha fazla üremede yer almak hakkı tanımak, kötüleri (iyileşirler ümidi ile) daha fazla değişime uğratmak uygulanan yöntemlerdendir. EB uygulamalarında kromozomun yapısının belirlenmesine koşut olarak kromozom üzerinde tanımlı bireyin çözüm olmak bakımından iyiliğini sayısal olarak veren bir fonksiyon yazılması gerekmektedir. Örnek olarak bir sırtçantası *(knapsack)* probleminde çözümün (bireyin) önerdiği toplam 'paha' değeri; bir gezgin satıcı *(TSP)* probleminde turun toplam uzunluğu ile azalan bir fonksiyon verilebilir.

Bu fonksiyonun tanımı her problem için sanıldığı kadar kolay olmayabilir. Örneğin verilmiş, rasgele biçimli bir alana lamba yerleştirerek aydınlatma probleminde kıymetli olanın tarifi çok kolay değildir. Birörnek (Homojen) ancak zayıf bir aydınlatma mı? yoksa bazı yerlerin karanlıkta kalması pahasına çok bölgenin güçlü aydınlatılması mı? Enerjiyi en aza indirmek önemli mi? Bozukluğa karşı güvence (bir lambanın bozulması sonucu bir bölgenin tamamen karanlıkta kalması) ne kadar önemli? Bu tür tasaları çoğaltmak olanaklıdır. EB uygulamalarında bu faktörler (ki bazen birbirleri ile çelişebileceklerdir de) bir ceza/ödül katkısı ile uygunluk fonksiyonunda yer alırlar.

EB nin tanıdığı bu esneklik çok ve heterojen parametreli problemlerin çözümünde kullanılmasını anlamlı kılmaktadır.

EB yöntemlerinin başarılı olarak kullanıldığı pek çok bilim ve mühendislik alanı vardır. Bunlardan bazılarını (örnekleri ile) sıralarsak:

En iyileme sayısal en iyileme (uçak kanadı tasarımı), birleşimsel en iyileme (baskılı devre tasarımı, işlik zamanlama problemleri)

Otomatik programlama Hücresel otomatların ve sıralama ağlarının tasarımı.

Makine öğrenmesi sınıflama ve öngörme görevleri (hava durumu öngörüsü, protein yapısı sınıflaması), ilişki ve yapı oluşturulması (sözdizim kurallarının bulunması)

Ekonomi Yaratıcı girişimin modellenmesi, pazarlık ve açık artırma stratejilerinin geliştirilmesi, ekonomik pazarların doğumunun modellenmesi.

Bağışıklık dizgesi canlı bedenin yaşam içerisindeki değişimi, doğal evrim süreçlerinde çoklu-gen ailelerinin varlığının bulunması.

Ekoloji Biyolojik silah yarışının, parazit-ana dizge eşevriminin, ortak yararlılığın ve kaynak akışının modellenmesi.

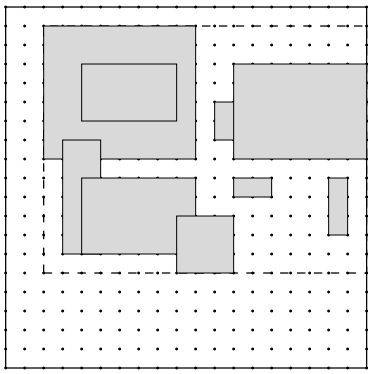
Evrim ve öğrenme Bireysel öğrenmenin türün evrimi ile etkileşimi.

Toplumsal dizgeler Çoklu etmen diszgelerinin iletişim ve işbirliği özelliklerinin gelişimi.

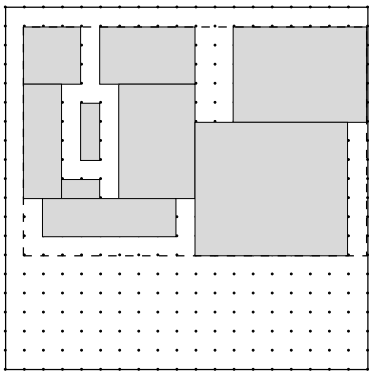
Son olarak, çeşitli EB yöntemlerinden GA ile bir Dünyevi problem çözümünün nasıl gerçekleştirildiğine daha yakından bakalım. Anlamayı kolaylaştırmak bakımından somut bir problem çözümünü adım adım inceleyeceğiz.

Ele alacağımız problem farklı boyutlardaki verilmiş dikdörtgenlerin tümünü alanı elden geldiğince küçük kılınmış bir dikdörtgen içine yerleştirmek olacak. Bu konuşlandırma sırasında hiç bir dikdörtgenin bir diğeri ile kısmen de olsa örtüşmemesi ve her dikdörtgenin bir kenarının düşey doğrultuda olması gerekiyor. Dolayısı ile bir dikdörtgeni ya verildiği üzere ya da 90^o çevrilmiş (devrik) olarak yerleştirebilmekteyiz. Dikdörtgenlerin boyutları tamsayı olarak verilmektedir.

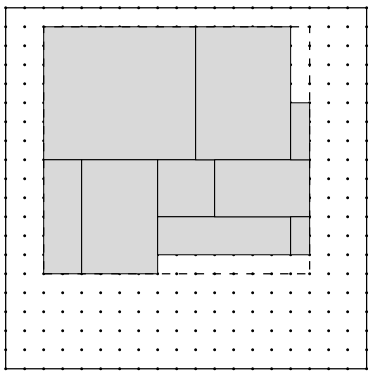
Aşağıda verildiği varsayılan bir küme dikdörtgenin rasgele bir yerleştirimini görmektesiniz. Bu yerleşim $13 \times 17 = 221$ $birim^2$ lik dikdörtgensel bir alan içinde kalmakla beraber bazı dikdörtgenlerin örtüşmesinden ötürü geçerli bir çözüm değildir.



Şimdi de geçerli bir yerleşim görelim:



Bu yerleşim $12 \times 17 = 221$ $birim^2$ lik toplam alana sahip geçerli bir yerleşimdir. Ancak aşağıdaki yerleşim probleme daha iyi bir çözüm önermektedir:



Bu yerleşim $13 \times 14 = 182$ *birim*^{2} lik bir alana sahiptir.

Şmdi adım adım bir problemin GA yöntemi ile çözülmesi için ne yapmamız gerektiğini irdeleyelim (*italik* yazı ile yazılmış bölümler örnek probleme ilişkindir; programlama dili özelindeki az sayıdaki ayrıntı C dilinde somutlaştırılıp verilmiştir)

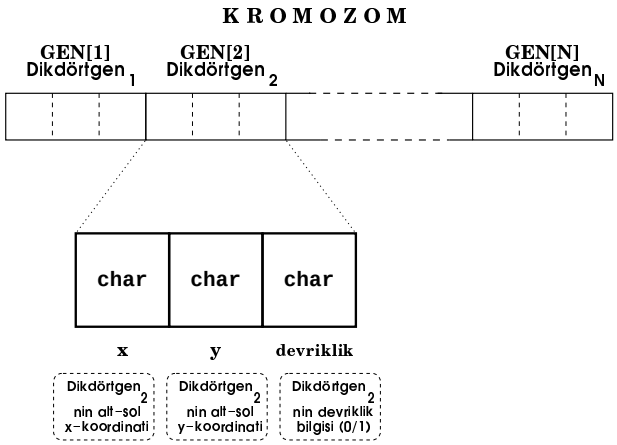
1. $$\text{Problem çözüm uzayı} \longmapsto \text{İkili sayı dizgileri uzayı}$$

   betimlemesi yapınız. Çoğunluk uygulamada ikili sayı dizgisinin boyu çözüm işlemi boyunca sabit tutulmaktadır.

   Örnek problemimiz için tüm bellekten yer almaların dinamik gerçekleştirildiği, i nci dizi öğesinin i nci dikdörtgenin sol-alt köşesinin yerleşim koordinatları ve dikdörtgenin devriklik bilgisini (örnegin `0`*:devrik değil,* `1`*:devrik) tutan 1-boyutlu bir dizi gösterimini seçeceğiz. her bir dizi öğesi (gen)*

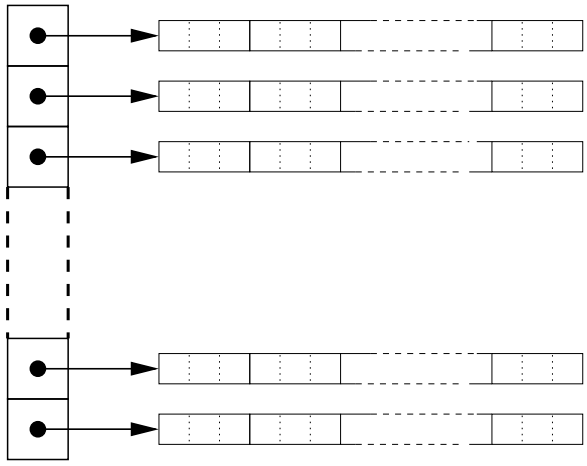
   ```
   struct gene
         { char x, y, orientation; };
   ```

   biciminde bir kayıt olacaktır. Diziye bir bütün olarak bakıldığında, problemimizin bir çözüm adayını kodlayabilen bir ikili sayı dizgisi algılanmaktadır.



2. Belirlenmiş türden dizgileri tutabilecek bir havuz oluşturunuz. Bu, genellikle elemanları dizgiler olan bir dizi değişkeni olarak gerçekleştirilir. Yeni kuşak oluşturulduktan sonra kısa bir süre için eski kuşağa da ihtiyaç olacaktır. Dolayısı ile havuz oluşturulurken yeni kuşak kromozamların konacağı ikinci bir havuz oluşturmak iyi bir tekniktir. *Örneğin C dilinde, havuz oluşturmak, öğelerinde yukarıda betimlenmiş 1-boyutlu dizilere birer imleç (pointer) tutulan bir dizi değişkeni ile kolayca sağlanabilir.*

H A V U Z



3. Düzgün bir rasgelelikle dizgilerin ilk değerlerini oluşturunuz.

4. $$Uygunluk \ : \ \text{İkili sayı dizgileri uzayı} \longmapsto \mathcal{R}$$

   olacak biçimde bir *Uygunluk* fonksiyonu tanımlayınız. $Uygunluk(s)$, s dizgisinin dünyevi problemin ne denli başarılı bir çözümünü kodladığının ölçütü olacaktır.

   Örnek problemimizde böyle bir ölçütü tanımlıyabilmek için bir çözümün iyiliğine ilişkin nitel bileşenler bulmamız mantıklı olacaktır. Bir kromozoma ilişkin aşağıdaki bileşenler önerilebilir:

   $Sayı_{saha\ dışı}$: *Tüm köşeleri yerleştirme ızgarasının içinde olmayan dikdörtgenlerin sayısı.*

   $Yüzölçüm_{KapsayanDD}$: *Saha dışı olmayan tüm dikdörtgenleri içine alan en küçük dikdörtgen alanın yüzölçümü.*

   $Yüzölçüm_{Örtüşen}$: *Saha dışı olmayan dikdörtgenlerin örtüşen alanların toplamı.*

   Şüphesiz iyi diyeceğimiz bir çözümde $Sayı_{saha\ dışı}$ *ve* $Yüzölçüm_{Örtüşen}$ *değerlerinin sıfır,* $Yüzölçüm_{KapsayanDD}$ *değerinin de küçük olması gerekir.*

   Ancak GA uygulamalarında önemli bir nokta bu fonksiyonun sürekliliğinin elden geldiğince gözetilmesidir. Sürekli bir Uygunluk fonksiyonu GA'ya çözümlerini iyileştirmesinde yol gösterici olacaktır.

   Bu bileşenleri tek bir toplam altında birleştirip her çözüm adayına ilişkin bir uygunluk değeri oluşturabiliriz:

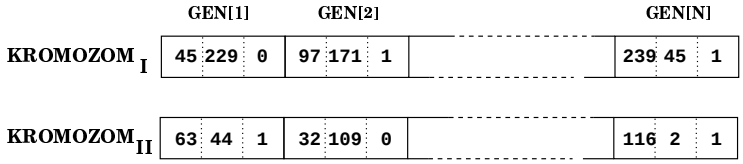
   $$C_1 \times Sayı_{saha\ dışı} + C_2 \times Yüzölçüm_{KapsayanDD} + C_3 \times Yüzölçüm_{Örtüşen}$$

   C_1, C_2 ve C_3 sabitlerinin pozitif olması durumunda yukarıda hesaplanan değerin sıfıra yakınlığı 'en uygunluk' anlamında olacaktır. Pratikte C_1 ve C_3 değerlerinin göreceli (C_2 ye göre) büyük seçilmesi saha dışılığı ve örtüşmeyi engelleyecektir.

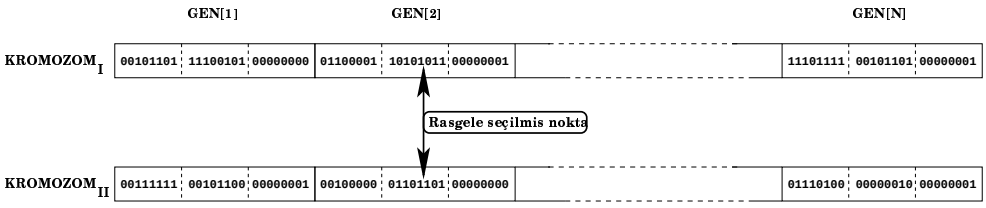
5. Havuzdaki tüm bireyleri (kromozomları) *Uygunluk*() fonksiyonu aracılığı ile değerlendirip sonuçları saklayınız. Genellikle bu, elemanları kayan noktalı sayılar olan, havuz değişkenine koşut, bir dizi değişkeninde gerçekleştirilir.

6. Havuzdan rasgele iki dizgi seçip, bu seçtiğiniz iki dizgiyi yine rasgele belirlediğiniz bir noktadan ikiye kesiniz. oluşan parçaları değiş tokuş ederek yeni iki dizgi elde ediniz, bu yeni dizgileri yukarıda açıklandığı üzere 'yeni' havuzda saklayınız. Bu işlemi havuzda çiftleşmemiş dizgi kalmayıncaya kadar sürdürünüz. Bazı GA uygulamalarında bir kromozomun çiftleşmede kullanılması uygunluk değeri ile koşut bir olasılık ile belirlenir. Yani uygunluk değeri yüksek bireyler daha fazla çiftleşmede yer alırken, uygunluk değeri düşük olanlar en fazlasıyla bir (bazen hiç) çiftleştirilmezler.

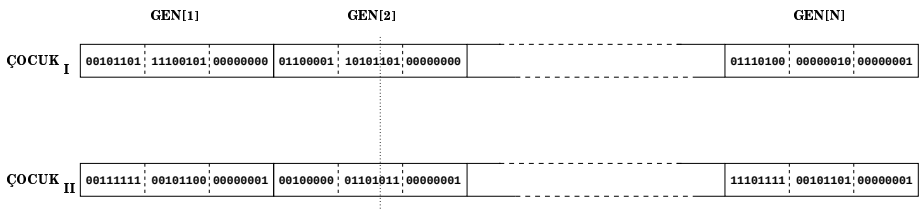
   Örneğimize ilişkin, çaprazlama işlemine daha yakından bakalım. Aşağıdaki iki kromozomun çaprazlanmak üzere eşleştirildiğini varsayalım:



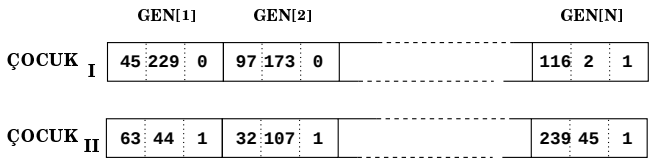
   Daha yakından (bit düzeyinde) bakar ve rasgele seçilmiş çaprazlama noktasını göz önüne alırsak:



   Görüldüğü üzere çaprazlama noktası büyük bir olasılıkla bazı ara bitlere denk düşecektir. Uygulama yazılımını oluşturulmasında bu durumun göz önünde bulundurulması ve çocukların sağlıklı üretilmesinin önlemlerinin alınması gereklidir. Aşağıda çaprazlama sonrası oluşan yeni kromozomları (çocukları) görülmektedir:



   Bit diziliminin karşılık geldiği sayıları belirtirsek:



Bu işlemin sonucunda, ata kromozomların bulunduğu havuza ek olarak bir de çocuk kromozonları içeren yeni bir havuz oluşturulmuş olmaktadır.

7. Mutasyon amacı ile yeni havuzdaki bireylerden bazılarını rasgele seçip rasgele bir bit'ini değiştiriniz. Bu çok yeğin bir işlem değildir. Havuzda bulunan toplam gen sayısı ($\langle Havuzdakikromozomsayısı\rangle \times \langle birkromozomdakigensayısı\rangle$) bakımından $\tilde{2}000$ gen başına bir bit'lik bir mutasyon yeğinliği sık kullanılan bir değerdir.

8. Yeni havuzdaki tüm dizgileri $Uygunluk()$ fonksiyonu aracılığı ile değerlendiriniz. Yeni havuzdaki dizgilerin (çocukların) sayısının eski havuzdaki (ata) dizgilerin sayısı ile aynı olduğuna dikkat çekmek isteriz.

9. Yeni oluşturulan havuzdaki bireylerden kimlerin yaşatılacağına kimlerinse öldürüleceğine karar vereceksiniz. Bunun için sık kullanılan bir yol *seçkinlik (elitism)* gözetmektir. Seçkinlik 'en uygun çözüm olma' anlamındadır. Eski havuzdaki en seçkinler (yani uygunluk değerleri en yüksek olanlar) arasından n tanesi bir kuşak daha yaşayabileceklerdir. Bu seçkin n birey dışındaki tüm eski havuz bireyleri öldürülüp yerlerine yeni havuzdakiler (en kötü n tanesi hariç olmak üzere) alınır. Uygulamalarda, n in toplam birey sayısına oranı %5 dolayındadır.

10. İşlemi durdurma ölçütü sağlanmamışsa (6) dan işleme devam et. İşlemi durdurma çoğunlukla belirli bir iyilikte çözüme varılmış olması, belirli sayıda kuşak oluşturulmuş olması veya uzun süreli olarak havuzda çözüm iyiliği bakımından bir değişiklik gözlenmemesi gibi bir ölçüte bağlanmaktadır.

Kaynakça

[1] T. Back, "Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms", Oxford, 1996.

[2] L. Davis, ed. "Genetic Algorithms and Simulated Annealing", Morgan Kaufmann, 1987.

[3] L. Davis, ed. "Handbook of Genetic Algorithm", Van Nostrand Reinhold, 1991.

[4] D. B. Fogel, "Evolutionary Computation: Toward a New Philosophy of Machine Intelligence", IEEE Press, 1995.

[5] D. E. Goldberg, "Genetic Algorithms in Search, Optimization, and Machine Learning", Addison-Wesley, 1989.

[6] J. H. Holland, "Adaptation in Natural and Artificial Systems", University of Michigan Press, Second edition, 1992.

[7] Z. Michalewicz, "Genetic Algorithms + Data Structures = Evolution Programs", Springer-Verlag, 1992.

[8] M. Mitchell, "An Introduction to Genetic Algorithms", MIT Press, 1997.

[9] G. Rawlins, ed. "Foundations of Genetic Algorithms", Morgan Kaufmann, 1991.

[10] H. P. Schwefel, "Evolution and Optimum Seeking", Wiley, 1995.

[11] D. ,Whitley, ed. "Foundations of Genetic Algorithms 2", Morgan Kaufmann, 1993.

[12] D. Whitley ve M. Vose, eds. "Foundations of Genetic Algorithms 3", Morgan Kaufmann, 1995.