Genetik Algoritma ve Tabu Arama Algoritması kullanarak N Kraliçe Problemini Çözme

Sezgisel Optimizasyon Dersi Proje Uygulaması 03.01.2020

Akın ÇAM
Bilgisayar Mühendisliği

Gebze Teknik Üniversitesi

Kocaeli, Turkey
aknncam@gmail.com

ÖZET

Bu makalede Genetik Algoritma ve Tabu Arama Algoritması kullanılarak n-Queen probleminin çözümü açıklanmaktadır. N-Queen problemi, Yapay Zeka araştırmacıları için zeka algoritmalarını uygulamak ve denemek için yaygın bir platform haline gelmiştir. Genetik Algoritmada sorunu çözmek için kullanılan kromozom bir problem çözümüdür ve Genetik Algoritmanın adımlarına bağlı olarak olası çözümler listelenmektedir. Tabu Arama algoritmasında komşuluk listesi oluşturulurken tabu listesi dikkate alınmaktadır. Bu sayede yeni komşuluk listeleri oluşturulup sonuca daha çabuk gidilmesi ve daha kaliteli sonuçların alınması sağlanmıştır.

Anahtar Kelimeler—n-Queen Problem, Genetic Algorithm, Tabu Arama Algoritması, GA,Sezgisel Optimizasyon.

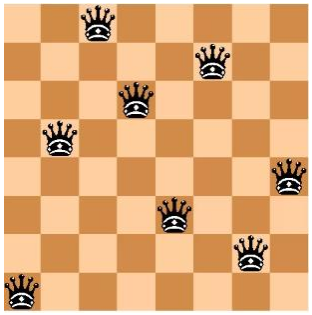
1 GİRİŞ

2 ALAKALI İŞLER

Yapılan araştırmalar sonucunda gösteriyor ki bu sorun üzerinde çalışmalar yapılmıştır. Ivica Martinjak[1] farklı sezgisel algoritmaları kullanarak karşılaştırmalar yapmıştır. Salabat Khan, Mohsin Bilal, M. Sharif, Malik Sajid, Rauf Baig[2] Ant Colony Algoritması ile bu problemi çözüme ulaştırmış fakat arama uzayı her bir vezir için n^2 olması algoritma üzerinde iyileştirmeler yapılması gerektiğini göstermektedir. 2012 yılında Farhad S. ve ark. ark. [3] DFS (Önce Derinlik Arama) ve BFS (Önce Genişlik Arama) tekniklerini kullanarak N-kraliçeleri çözmek için yeni bir çözüm önermişlerdir.

3 N-QUEEN PROBLEM

N-queens problemi 1850'de Carl Gauss tarafından ortaya atılmış ve onlarca yıldır bilim adamları tarafından incelenmiştir. Daha sonra 1948 yılında profesyonel satranç oyuncusu olan Max Bezzel tarafından ortaya atılmıştır. Yıllar içerisinde Gauss gibi önemli matematikçiler tarafından incelenen problemi ilk çözümü 1950 yılında Franz Nauck tarafından sunuldu. Ayrıca Franz Nauck problemi n*n lik tahta üzerinde genelleştirerek n vezir problemi haline de getirmiştir. n-Queen problemde n*n satranç tahtası üzerinde n sayıdaki kraliçenin birbirini yemeyecek şekilde konumlandırılmasıdır. (Aynı satır, aynı sütun veya aynı çarpraz sırada iki veya fazla kraliçe bulunamaz).



Şekil-1 8*8 kraliçe problemi için bir çözüm

VEZİRLER	Çözüm sayısı
1	1
2	0
3	0
4	2
5	10
6	4
7	40
8	92
9	352
10	724
11	2.680
12	14.200
13	73.712
14	365.596
15	2.279.184
16	14.772.512
17	95.815.104
18	666.090.624
19	4.968.057.848

Şekil-2 n*n kraliçe problemi için çözüm sayıları

4 GENETİK ALGORİTMA

GA popüler meta-sezgisel algoritmalardan biridir, kromozom popülasyonuna göre uzay çözümünü temsil eder ve her kromozom problem için bir çözüm olarak çalışır. GA, fitness fonksiyonuna ve ayrıca algoritmanın yapısına bağlı olarak en uygun çözümü verir. Algoritma, optimum çözüme (durma koşulu) ulaşıncaya kadar kromozomun geçişini ve mutasyon işlemlerini tekrarlar.

A. Genetik Algoritma Adımları

Adım 1:

Kullanıcıdan alınan popülasyon sayısı kadar kromozom üretilir. Bu kromozomlar üretilirken kraliçeler karıştırılır ve

popülasyonda farklı elemanların bulunması sağlanır. Her eleman popülasyona eklendikten sonra popülasyondaki elemanın çözüm karmaşıklığı hesaplanır. Çözüm karmaşıklığı bir kraliçenin tüm komşulukları(alt-üst-sağ-sol-çapraz-alanlar) kontrol edilir ve herhangi bir kraliçe komşusu var ise karmaşıklık arttırılır.

Şekil-3 n*n kraliçe problemi için karmaşıklık hesaplama için kullanılan komşuluklar.

Adım 2:

Bitme koşulu sağlandığı sürece(Algoritma çalışma iterasyonu) Fitness değeri hesaplanır. Fitness hesabına bir populasyondaki en büyük ve en küçük karmaşıklık bulunarak başlanır. Daha sonra bu değerin ortalaması alınır. Populasyondaki her elemanın bu ana karmaşıklığa göre fitness değeri hesaplanır.

Fitness = (max karmaşıklık – kromozom karmaşıklığı) *100/ ortalama karmaşıklık

Adım 3:

Fitness değerleri hesaplanan komşuluklara selection işlemi uygulanır. Burada roulette selection yöntemi şöyle uygulanmıştır: Toplam fitness değeri bulunur, seçilme olasılığı işlenir, Alınan bi random değer boyunca tıpkı bir çarkıfelek gibi seçilme olasılıkları toplanarak eşleşen index populasyona eklenir.

Adım 4:

Seçme işleminden sonra crossover ve mutasyon işlemleri gerçekleştirilir. Kullanıcıdan alınan offSpring değeri boyunca her seferinde bir random değer mating_probability değerini sağlayıp sağlamadığına bakılır. Sağlıyorsa iki tane rassal olarak kromozom alınır ve 2 point crossover yapılır. Daha sonra rassal değer mutasyon değerini sağlıyorsa kullanıcıdan alınan mutasyon sayısının rassal olarak seçilmesi sonucu mutasyon işlemi gerçekleştirilir.

Adım 5:

Burada populasyondan rastgele 20 eleman çıkarılır ve değerler ilklendirilir.

5 Tabu Arama Algoritması

Tabu Arama Algoritması bir sezgisel optimizasyon algoritmasıdır. N-queen probleminde popülasyon oluşturulurken tabu olan komşular popülasyon listesine eklenmez ve bu sayede algoritmanın aynı çözüm uzayında tekrar tekrar çalışması önlenmiş olur.

B. Tabu Arama Algoritması Adımları

Adım 1:

İlk olarak bir çözüm oluşturulur. Bu çözümün fitness değeri hesaplanır ve en iyi çözüm olarak kaydedilir.

Adım 2:

Algoritma maksimum iterasyon sayısı kadar devam edecektir. Bu adımda bir popülasyon oluşturulur. Populasyona elemanlar eklenirken tabu listesinde olup olmadığına bakılır. Eğer rassal çözüm tabu ise popülasyona eklenmez değilse hem popülasyona hem tabu listesine eklenir.

Adım 3:

Popülasyondaki her eleman elemanın fitness değeri hesaplanır. Fitness değeri bir kraliçenin tüm komşulukları(alt-üst-sağ-sol-çapraz-alanlar) kontrol edilir ve herhangi bir kraliçe komşusu var ise fitness değeri artar fakat bu daha az kaliteli anlamına gelmektedir.

Adım 4:

Populasyondaki elemanlar şu ana kadar ki en iyi çözümle karşılaştırılır ve iyi bir değer bulunmuşsa bu değer güncellenir.

Adım 5:

Fitness değeri 0 ise yani çözümün hiçbir karmaşıklığı yoksa algoritma amacına ulaşmış ve çözüm bulunmuştur.

Adım 6:

Her iterasyonda tabu listesinin yarısı silinir.

6. Sonuçlar ve Tartışma

Tabu Arama Algoritması için yapılan testler sonucunda elde edilen sonuçlar ve yapılan çıkarımlar şöyledir:

Şekil 4 de görüldüğü gibi tahta sayısı ve popülasyon sayısı sabit olduğunda tabu listesi sayısının popülasyondan büyük olması durumunda daha çabuk çözüme ulaşıldığı görülmüştür. Fakat Tabu listesinin boyutunun çok artması algoritmayı sonuca ulaştırmakta fakat çalışma zamanını arttırmaktadır. Sonuç olarak tabu listesi sayısı popülasyon sayısından büyük fakat çok uç noktalarda olmamalıdır.

Şekil 5 teki grafik alınırsa bir çözüm için popülasyon sayısının artması algoritma hızını bir yere kadar doğru orantılı etkilemektedir. Fakat Çok uç noktalarda olması algoritma çalışma zamanı için kötü yönde ilerlemektedir. Populasyon sayısı ortalama olarak tahta sayısının karesi civarında olması optimal sonuca hızlı ulaşmak için iyi olacaktır.

Şekil 6 gösteriyor ki popülasyon sayısının artması algoritmayı yavaşlatmakta fakat mutlak çözüm için yüksek olması gerekebilir.

Şekil 7 ye baktığımızda şu sonucu çıkarabiliriz. Populasyon sayısı arttıkça algoritmanın çalışma süresi artmaktadır. Fakat burada dikkat edilmesi gereken nokta şudur: Tahta boyutu arttıkça kraliçelerin yerleşeceği permutasyon sayısı artmaktadır bu neden çözüme ulaşılmak için iterasyon sayısının yüksek olması gerekmektedir.

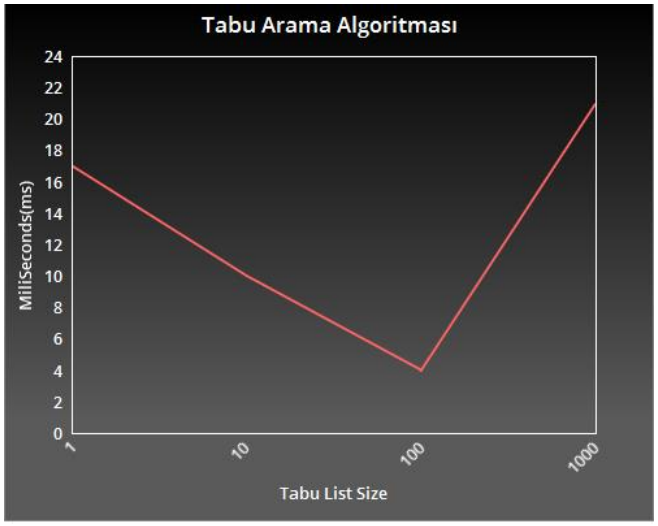
Genetik Algoritması için yapılan testler sonucunda elde edilen sonuçlar ve yapılan çıkarımlar şöyledir:

İterasyon sayısı ne kadar artarsa artsın eğer popülasyon sayısı tahta sayısına uygun bir değer değil ise çözüm elde edilememektedir. Çözüm elde etmek için popülasyon sayısının sadece artması yetmemektedir ve algoritma iterasyon sayısının da arttırılması gerekmektedir. Tahta sayısı 10 algoritma iterasyon sayısı 10 sabit olmak üzere popülasyon sayısı 10 100 1000 vb gibi değerler 10 ar defa

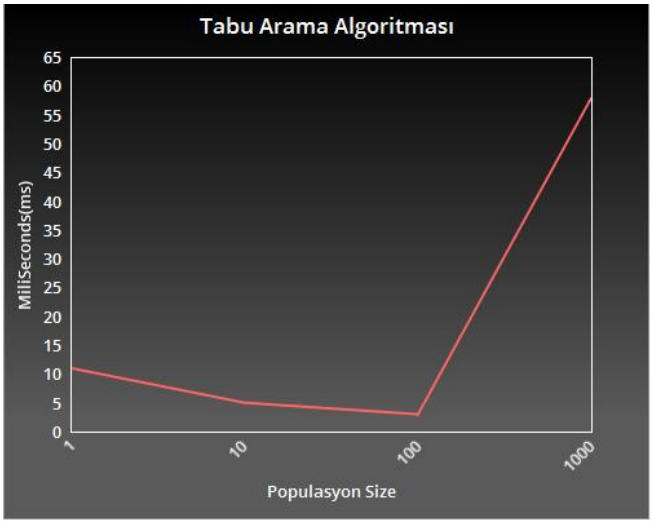
çalıştırılarak test edilmiş fakat sonuca ulaşılamamıştır. Bu sebeple iterasyon sayısı ve popülasyon sayısı birbiriyle orantılı ve board sayısının karesinden fazla olmalıdır. Bu durum tahta sayısı 10 un üzeri için geçerlidir.

OffSpring değerinin düşük olması algoritmayı çözümden uzaklaştırmaktadır. Yukarıdaki değerlerin artması ve bunlarla birlikte offspring değerinin de artması mutlak sonuç için ihtiyaçtır. Şekil 8 de görüldüğü gibi offspring sayısının artması algoritmanın mutlak sonuca ulaşması için gereklidir.

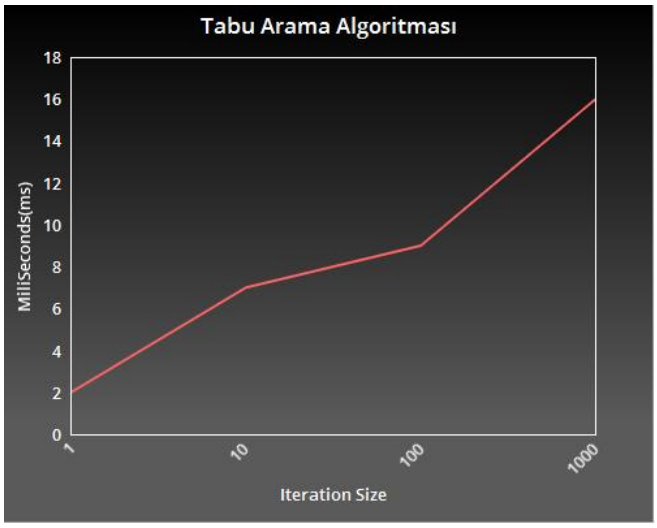
Mutasyon oranının artması her zaman mutlak sonuca götürmesede çözüm bulunma ihtimalini arttırmaktadır. Mutasyon sayısı 0.001 ve 0.1 ile test edilmiş ve 0.1 in ortalama iki çalışmadan birinde sonuca ulaştırdığı görülmüştür.



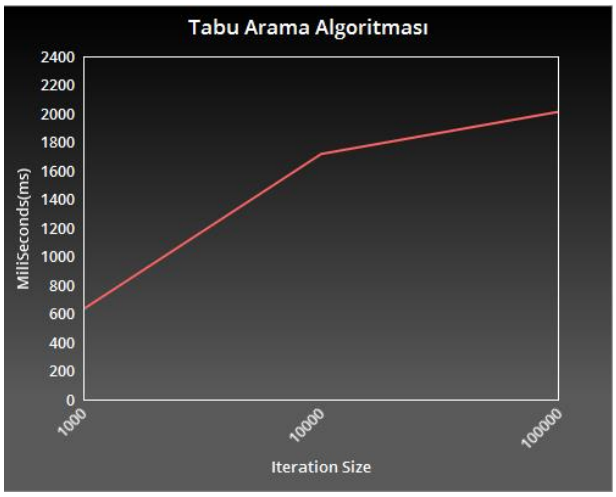
Şekil-4(Tahta Boyutu = 8, populasyon sayısı = 10, iterasyon sayısı = 100)



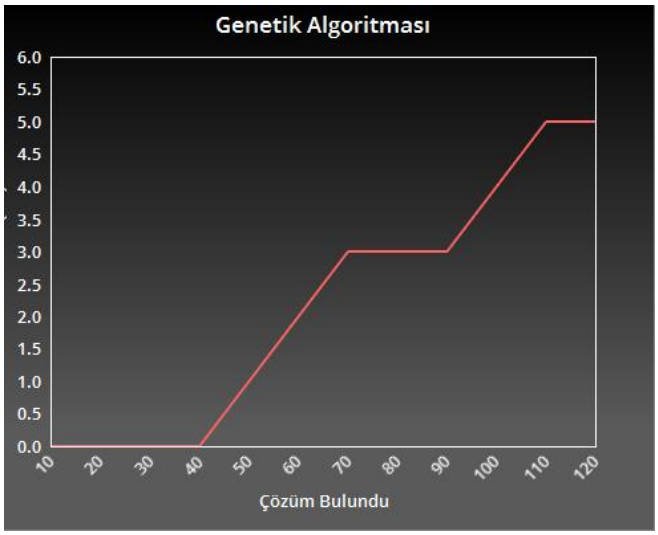
Şekil-5 (Tahta Boyutu = 8, tabu sayısı = 10, iterasyon sayısı = 100)



Şekil-6(Tahta Boyutu = 8, tabu sayısı = 10, populasyon sayısı = 10)



Şekil-7(Tahta Boyutu = 12 tabu sayısı = 300, populasyon sayısı = 150)



Şekil-8(Tahta Boyutu = 12 tabu sayısı = 300, populasyon sayısı = 150, Offsprin sayısına göre sonuca ulaşma dereceleri(her biri 10 ar defa test edilmiştir.)

REFERENCES

[1] Comparison of Heuristic Algorithms for the N-Queen Problem Ivica Martinjak

[2] Solution of n-Queen Problem Using ACOSalabat Khan, Mohsin Bilal, M. Sharif, Malik Sajid, Rauf Baig

[3] Farhad S., Bahareh S. and G. Feyzipour, "A New Solution for N-Queens Problem using Blind Approaches: DFS and BFS Algorithms", International Journal of Computer Applications (0975 – 8887) Volume 53– No.1, September 2012.