T.C.
SÜLEYMAN DEMİREL ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

SATRANÇ OYUNU DURUM UZAYININ SEZGİSEL MODELLENMESİ VE GLOBAL OPTİMİZASYON YÖNTEMLERİ İLE ÇÖZÜMÜ

Melike ŞİŞECİ ÇEŞMELİ

Danışman
Doç. Dr. Bayram CETİŞLİ

DOKTORA TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
ISPARTA - 2016

TEZ ONAYI

Melike ŞİŞECİ ÇEŞMELİ tarafından hazırlanan **"Satranç Oyunu Durum Uzayının Sezgisel Modellenmesi ve Global Optimizasyon Yöntemleri ile Çözümü"** adlı tez çalışması aşağıdaki jüri üyeleri önünde Süleyman Demirel Üniversitesi Fen Bilimleri Enstitüsü **Bilgisayar Mühendisliği Anabilim Dalı**'nda **DOKTORA TEZİ** olarak başarı ile savunulmuştur.

Danışman	**Doç. Dr. Bayram CETİŞLİ** Süleyman Demirel Üniversitesi	
Jüri Üyesi	**Prof. Dr. Akif KUTLU** Süleyman Demirel Üniversitesi	
Jüri Üyesi	**Doç. Dr. Ecir Uğur KÜÇÜKSİLLE** Süleyman Demirel Üniversitesi	
Jüri Üyesi	**Yrd. Doç. Dr. Göksel ASLAN** Mehmet Akif Ersoy Üniversitesi	
Jüri Üyesi	**Yrd. Doç. Dr. Sedat METLEK** Mehmet Akif Ersoy Üniversitesi	

Enstitü Müdürü **Doç. Dr. Yasin TUNCER**

TAAHHÜTNAME

Bu tezin akademik ve etik kurallara uygun olarak yazıldığını ve kullanılan tüm literatür bilgilerinin referans gösterilerek tezde yer aldığını beyan ederim.

Melike ŞİŞECİ ÇEŞMELİ

İÇİNDEKİLER

Sayfa

İÇİNDEKİLER i
ÖZET ii
ABSTRACT iii
TEŞEKKÜR iv
ŞEKİLLER DİZİNİ v
ÇİZELGELER DİZİNİ vi
SİMGELER VE KISALTMALAR DİZİNİ vii
1. GİRİŞ 1
2. KAYNAK ÖZETLERİ 7
3. MATERYAL VE YÖNTEM 17
3.1. ELO Derece Sistemi 17
3.2. Amaç Fonksiyonu 17
3.3. Arama Mekanizması 23
3.3.1. Minimaks yöntemi 24
3.3.2. Alfa-Beta (α-β) budama 26
3.3.3. Kümeleme tabanlı global optimizasyon yöntemi 30
3.3.4. Parçacık sürü optimizasyonu algoritması 34
3.3.5. Geliştirilmiş yarasa algoritması 35
3.3.6. Yapay arı kolonisi algoritması 36
3.3.7. Genetik algoritma 37
3.4. Geliştirilen Program ve Arayüz 39
4. ARAŞTIRMA BULGULARI VE TARTIŞMA 42
4.1. Chess Veritabanı Üzerinde Yapılan Çalışmalar 42
4.2. SCHACKNYTT Veritabanı Üzerinde Yapılan Çalışmalar 56
4.3. Sezgisel Yöntemlerin Birbirleriyle Yaptığı Maçların Karşılaştırılması 60
5. SONUÇ VE ÖNERİLER 66
KAYNAKLAR 68
ÖZGEÇMİŞ 73

ÖZET

Doktora Tezi

SATRANÇ OYUNU DURUM UZAYININ SEZGİSEL MODELLENMESİ VE GLOBAL OPTİMİZASYON YÖNTEMLERİ İLE ÇÖZÜMÜ

Melike ŞİŞECİ ÇEŞMELİ

Süleyman Demirel Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. Bayram CETİŞLİ

Satranç oyunu, taşlarının yapabileceği hamle olasılıkları ve diğer kuralları düşünüldüğünde basit gibi algılanıp çabuk öğrenilebilse de, ileriki hamleler düşünülüp yapılabilecek hamlelere ait çözüm uzayı oluşturulduğunda, oyunun zorluğu ve stratejik yönü daha iyi anlaşılabilmektedir. Optimizasyon ise mevcut durumlar arasında en iyi çözüm olarak tanımlanmakta olup, bu çalışmada satranç oyununa ait amaç fonksiyonu için kullanılmıştır.

Satranç oyununda derinliğe göre milyonları bulabilen olası hamlelerin taranabilme zorluğu dolayısıyla, belli bir derinlikten sonra klasik arama algoritmaları ile çözüme ulaşılamamaktadır. Bu çalışmada, sezgisel algoritmaların gücü ile bu sorun aşılmaya çalışılmıştır. Sezgisel algoritmalar gücünü tüm uzayı taramak yerine umut vaat eden bölgelere odaklanıp en iyi sonucu aramalarıyla elde etmektedirler.

Bu tez çalışmasında, satranç oyununda daha derinlerde yer alan çözümlerin aranmasında sezgisel yöntemlerden yararlanılıp, kümeleme tabanlı global optimizasyon, yapay arı kolonisi vb. popüler yöntemler açıklanmıştır. Bu yöntemlerin kendi aralarında yaptıkları karşılaşmalar, SCHACKNYTT (chessnews)'de yayınlanan ünlü satranç oyuncularının oynadığı popüler oyunlardaki stratejik hamlelerin tahmini ve çeşitli oyun sonu hamlelerinin tahmini baz alınıp, yöntemlerin performansları elde ettikleri puanlara göre test edilip karşılaştırılmıştır. Ayrıca geliştirilen yazılım, kullanıcı arabirimi sayesinde insan ve piyasadaki diğer satranç motorlarına karşı da oyun oynayabilmektedir. Deneysel çalışmalarda elde edilen sonuçlar, satranç oyununda arama yöntemi olarak sezgisel yöntemlerin kullanılmasının avantajlarını vurgularken, daha derinlerde arama yapılabilmesi olanağını da ortaya koymaktadır.

Anahtar Kelimeler: Satranç oyunu, Arama algoritmaları, Sezgisel arama, Kümeleme tabanlı global optimizasyon, Oyun teorisi, Oyun ağacı.

2016, 75 sayfa

ABSTRACT

Ph.D. Thesis

HEURISTIC MODELING OF STATE SPACE OF CHESS GAME AND GLOBAL OPTIMIZATION SOLUTIONS

Melike ŞİŞECİ ÇEŞMELİ

Süleyman Demirel University
Graduate School of Applied and Natural Sciences
Department of Computer Engineering

Supervisor: Assoc. Prof. Dr. Bayram CETİŞLİ

Chess game, despite perceived to be as simple and learned quickly, in terms of the possibilities that can move the pieces and when the other rules are considered, the difficulty of the game and its strategic aspect can be better understood, when further moves are considered and the solution space of the for coming moves is created. Optimization is defined as the best solution in the present circumstances, and used for the objective function of the chess game in this study.

Due to the difficulty of screening the possible moves which according to the depth of the chess game can be millions in numbers, after a certain depth, a solution can not be reached by conventional search algorithms. In this study we tried to overcome this problem with heuristic algorithms power. Power of heuristic algorithms are achieved by focusing on the promising regions and searching for the best results instead of scanning the entire space.

In this thesis study, in the aspect of investigation of in depth solutions to the chess game, benefited from the intuitive method, global optimization based on clustering, and popular methods such as artificial bee colony et al. are described. These methods that they encounter among themselves, are based on the estimation of strategic moves in the popular games that the famous chess players took part in as published in SCHACKNYTT (chess news), and estimation of various endgame moves, the performances of the methods are tested according to the points earned and compared. Furthermore, the developed software, thanks to the user interface, can play against people and other chess engines in the market. The results obtained in experimental studies, emphasizing the advantages of the use of heuristic method as a searching tool as well as revealing the possibility of deeper search.

Keywords: Chess game, Search algorithms, Heuristic search, Global optimization based on clustering, Game theory, Game tree.

2016, 75 pages

TEŞEKKÜR

Doktora çalışmalarım süresince desteklerini ve katkılarını asla unutamayacağım ve kendilerini her zaman minnetle anacağım hocalarıma, aileme ve arkadaşlarıma öncelikle şükranlarımı sunarım. Yüksek Lisans çalışmalarımda olduğu gibi, doktora çalışmalarımda da güveni, bilgisi ve zamanını asla esirgemeyen değerli danışman hocam Doç. Dr. Bayram CETİŞLİ'ye sonsuz teşekkür borçluyum. Tüm çalışmalarım süresince bana ışık tutan, ihtiyacım olan bilgilerini ve zamanlarını benden esirgemeyen değerli hocalarım Prof. Dr. Akif KUTLU'ya, Doç.Dr. Ecir Uğur KÜÇÜKSİLLE'ye de teşekkür ederim.

Doktora tezimin birçok aşamasında bilgisi, düzeltmeleri, tavsiyeleri ile yaptığı katkılarından dolayı değerli arkadaşım Uzman İhsan PENÇE'ye teşekkür ederim.

Birlikte ders aldığım diğer tüm arkadaşlarıma da yardım ve desteklerinden ötürü teşekkür ederim.

Çalışmalarım süresince beni sürekli destekleyen ve cesaretlendiren sevgili eşim Hasan ÇEŞMELİ'ye, biricik kızım İkra Gökçe ÇEŞMELİ'ye annem Fatıma GÜLER'e, babam Mehmet ŞİŞECİ'ye ve kardeşlerim Nurgül ve Hakan ŞİŞECİ'ye en derin sevgi ve teşekkürlerimi iletiyorum.

Son olarak, burada belirtmemiş olabileceğim ancak bu çalışmada az veya çok katkısı olan herkese teşekkürü bir borç bilirim. Sevgi, saygı ve minnetlerimle…

Melike ŞİŞECİ ÇEŞMELİ
ISPARTA, 2016

ŞEKİLLER DİZİNİ

Sayfa

Şekil 1.1. Satranç motoruna ait temel mimari 3
Şekil 3.1. Örnek satranç dizilimi 1 20
Şekil 3.2. Örnek satranç dizilimi 2 21
Şekil 3.3. Örnek satranç dizilimi 3 22
Şekil 3.4. Minimaks algoritmasının Ağaç Arama algoritmasındaki kullanımı 25
Şekil 3.5. Minimaks'ın satranç oyunundaki yapısının akış diyagramı 26
Şekil 3.6. Basit bir α-β problemi 27
Şekil 3.7. Basit bir α-β problemi çözümü 28
Şekil 3.8. α-β satranç oyunundaki yapısının akış diyagramı 29
Şekil 3.9. Popülasyonların kümeleme işlemi ile küme merkezlerinin tespiti 31
Şekil 3.10. Popülasyonların parabolik eğri uydurma ile minimum ve maksimum noktalarının tespiti 32
Şekil 3.11. KTGO'nun satranç oyunundaki yapısının akış diyagramı 33
Şekil 3.12. Geliştirilen programa ait arayüz 40
Şekil 4.1. Chess veritabanında yer alan 27. oyun 43
Şekil 4.2. Chess veritabanında yer alan 1327. oyun 44
Şekil 4.3. Chess veritabanında yer alan 2015. oyun 45
Şekil 4.4. Chess veritabanında yer alan 2756. oyun 46
Şekil 4.5. Chess veritabanında yer alan 2825. oyun 47
Şekil 4.6. Chess veritabanında yer alan 2845. oyun 48
Şekil 4.7. Chess veritabanında yer alan 2873. oyun 49
Şekil 4.8. Chess veritabanında yer alan 2900. oyun 50
Şekil 4.9. Chess veritabanında yer alan 3000. oyun 51
Şekil 4.10. Chess veritabanında yer alan 3047. oyun 52
Şekil 4.11. Chess veritabanında yer alan 3098. oyun 53
Şekil 4.12. Chess veritabanında yer alan 3140. oyun 54
Şekil 4.13. Chess veritabanında yer alan 3300. oyun 55
Şekil 4.14. SCHACKNYTT veritabanında yer alan 16 oyun 57

ÇİZELGELER DİZİNİ

Sayfa

Çizelge 3.1. TSCP'nin amaç fonksiyonuna ait parça ve pozisyon değerleri 17
Çizelge 3.2. Amaç fonksiyonundaki özniteliklere ait katsayılar 22
Çizelge 3.2. Amaç fonksiyonundaki özniteliklere ait katsayılar (Devam) 23
Çizelge 4.1. 27. Oyuna ait puan ve süre değerleri 43
Çizelge 4.2. 1327. Oyuna ait puan ve süre değerleri 44
Çizelge 4.3. 2015. Oyuna ait puan ve süre değerleri 45
Çizelge 4.4. 2756. Oyuna ait puan ve süre değerleri 46
Çizelge 4.5. 2825. Oyuna ait puan ve süre değerleri 47
Çizelge 4.6. 2845. Oyuna ait puan ve süre değerleri 48
Çizelge 4.7. 2873. Oyuna ait puan ve süre değerleri 49
Çizelge 4.8. 2900. Oyuna ait puan ve süre değerleri 50
Çizelge 4.9. 3000. Oyuna ait puan ve süre değerleri 51
Çizelge 4.10. 3047. Oyuna ait puan ve süre değerleri 52
Çizelge 4.11. 3098. Oyuna ait puan ve süre değerleri 53
Çizelge 4.12. 3140. Oyuna ait puan ve süre değerleri 54
Çizelge 4.13. 3300. Oyuna ait puan ve süre değerleri 55
Çizelge 4.14. SCHACKNYTT oyunu sonuçları 57
Çizelge 4.15. Ünlü satranç oyun motorlarının ve programdaki yöntemlerin SCHACKNYTT'den aldıkları puanlar 58
Çizelge 4.16. Ünlü satranç oyun motorlarının ve programda yer alan yöntemlerin ilk 8 SCHACKNYTT durumlarından aldıkları puanlar 59
Çizelge 4.17. Ünlü satranç oyun motorlarının ve programda yer alan yöntemlerin son 8 SCHACKNYTT durumlarından aldıkları puanlar 59
Çizelge 4.17. Ünlü satranç oyun motorlarının ve programda yer alan yöntemlerin son 8 SCHACKNYTT durumlarından aldıkları puanlar (Devam) 60
Çizelge 4.18. Yöntemlerin sıfır oyunda birbirleriyle karşılaştırılması 61
Çizelge 4.18. Yöntemlerin sıfır oyunda birbirleriyle karşılaştırılması (Devam) 62
Çizelge 4.19. Yöntemlerin sıfır oyunda kazandıkları oyun sayısı 62
Çizelge 4.20. 2832. Oyunda siyah oyuncuya ait sezgisel algoritmaların birim iterasyon süre değerleri 63
Çizelge 4.20. 2832. Oyunda siyah oyuncuya ait sezgisel algoritmaların birim iterasyon süre değerleri (Devam) 64
Çizelge 4.21. 2832. Oyunda beyaz oyuncuya ait sezgisel algoritmaların birim iterasyon süre değerleri 64

SİMGELER VE KISALTMALAR DİZİNİ

DGA	Diferansiyel gelişim algoritması
FIDE	Dünya satranç federasyonu
GA	Genetik algoritma
GYA	Geliştirilmiş yarasa algoritması
KTGO	Kümeleme tabanlı global optimizasyon
ŞKŞ	Şah kale şah
YA	Yarasa algoritması
YAK	Yapay arı kolonisi
YSA	Yapay sinir ağı

1. GİRİŞ

Bilgisayar, günümüzde hem donanımsal, hem de yazılımsal olarak sürekli gelişen bir dinamiktir. Donanımın yeterliliği çerçevesinde artık bilgisayarla geleneksel çözüm yöntemlerinin yanı sıra karmaşık problemlerin çözümü için sezgisel yöntemler de kullanılmaya başlamıştır.

Optimizasyon, bir amaç fonksiyonunun verilen kısıtlar dâhilinde minimum yada maksimum değerini veren değişken değerlerinin bulunması işlemi olarak tanımlanmaktadır (Edgar ve Himmelblau, 1989). Optimizasyon problemlerinin çözümünde her zaman en iyi çözüm garantilenmese de, genellikle yeterli çözüm bulunabilmektedir. Gerçek hayat problemlerinin birçoğunda birden fazla çözüm bulunmakta olup bu çözümler optimizasyon işlemleriyle gerçekleştirilir. Problemin tek bir parametre listesinin ve çözümünün olduğu durumlarda optimizasyon işlemi uygulanamamaktadır (Çakar, 2009). Yapay zeka optimizasyon algoritmaları her problemde aynı şekilde başarılı olamayabilirler. Başarıyı garantilemek veya artırmak için problemin yapısını ve kısıtlarını iyi anlamak, ona göre bir teknik seçmek gerekmektedir.

Yapay zekâ algoritmaları, bilgisayarlı masa oyunlarında 1940'li yıllardan bu yana uygulanmaktadır. Masa oyunları rekabetçi, bilişsel öğrenme ve dinamik yapısı dolayısıyla hesaplamalı zekâ teorileri ve algoritmalar için ideal bir alandır. Doğal evrim de bir oyun olarak kabul edilebilir (Singh vd., 2011).

Oyun ise; oyuncu kümesini, bu oyuncular tarafından kullanılabilir hamle kümesini ve her bir stratejinin kombinasyonundan edinilen sonuçları içerir. Oyunun belli bir adımının çözümünde, bireyin başarısının diğerlerinin seçimlerine dayalı olduğu hamleler yaptığı bazı stratejik durumların matematiksel ifadelerinin çözümü kullanılmaktadır (Myerson, 1991).

Satranç tarihi incelendiğinde, oyun M.Ö. 6. yüzyılda Hindistan'da ortaya çıktığı M.S. 10. yüzyıla gelindiğinde ise tüm Asya, Ortadoğu ve Avrupa'ya yayıldığı görülmektedir. 15. yüzyıldan itibaren Avrupa'da soylular arasında çok popüler

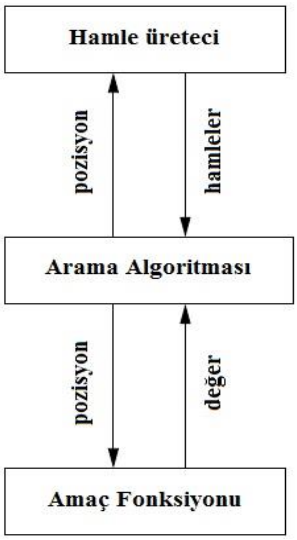
hale gelip Kraliyet oyunu olarak anılmaya başlanmıştır. Kuralları ve dizilişleri zaman içerisinde çeşitli değişiklikler göstermiş olup 19. yüzyılda bugünkü standart halini almıştır (Vikipedi, 2015). Satranç, iki farklı renkli ve 8X8'lik kare bir platform üzerinde oynanan bir oyundur. Oyunculardan birinin beyaz, diğerinin siyah olan 16'şar adet taşı vardır. Bu taşlar birer adet şah ve vezir, ikişer adet at, kale, fil ve 8 adet piyondan oluşmaktadır. Bu taşlar oyun platformu üzerinde belli bir düzen içinde yerleşmişlerdir. Oyuna beyaz taşa sahip oyuncu başlar ve oyun boyunca her taş kendi hamle yeteneğine göre oyuncular tarafından sırayla hareket ettirilir. Her iki oyuncunun amacı rakip şahı mat etmektir. Her iki oyuncu için mat durumu mümkün değilse oyun berabere biter. Buna pat ismi verilmiştir.

1947 yılında Alan Turing (Turing vd., 1953) satranç oynayan ilk programı oluşturduktan sonra, satranç oyununun ne kadar zor bir optimizasyon problemi olduğu daha da iyi anlaşılmıştır. 1950'li yıllarda bilgisayarın satranç oyununu oynayabilmesi için, satranç ustalarından daha fazla derine bakabilecekleri şekilde programlanmaları gerektiği düşünülmekteydi. Ancak satranç oyunu başlangıcında 10^{123} pozisyon incelenmelidir ve günümüzdeki hiçbir bilgisayar bunu hesaplayabilecek güçte değildir (Gültekin, 2006). Günümüzde halen daha satranç programının çözümünü iyileştirmek için çalışmalar yapılmakla birlikte, çözüm uzayının aşırı büyük olması sebebiyle günümüz bilgisayarları çözüm uzayında belli bir derinliğin altına inememektedir.

Alan Turing'den daha sonra Claude Shannon (Shannon, 1950) tarafından iki farklı strateji satranç programına dâhil edilmiştir. Bu stratejilerden ilki arama ağacının belli bir derinliğinde bütün olası hamlelerin değerlendirilmesi iken, diğer strateji satranç bilgisini kullanarak daha derinlerdeki en çok umut vaat eden hamlelerin tespitidir. 1950 yılında çok basit seviyede satranç oynayan programlar, 1970 yıllarında sezgisel seçimler kullanarak en iyi hamleleri aramaya başlamışlardır. 1997 yılında ise IBM firmasına ait Deep Blue isimli bilgisayar dünya satranç şampiyonu Garry Kasparov'u yenmiştir. Deep Blue saniyede 200 milyon pozisyon hesaplayabilmektedir (Vázquez-Fernández vd., 2012).

Genellikle bir satranç motorunun gücü; hamle üreticisinin etkinliği, arama ağacının derinine inme mesafesi ve pozisyonların hesaplanmasını sağlayan fonksiyon ile hesaplanır.

Amaç fonksiyonu kullanan satranç motorları çoğunlukla arama algoritması, hamle üreteci ve amaç fonksiyonu olmak üzere 3 ana bileşenden oluşmaktadır. Şekil 1.1'de satranç motoruna ait temel mimari görülmektedir (Vázquez-Fernández ve Coello, 2013).



Şekil 1.1. Satranç motoruna ait temel mimari (Vázquez-Fernández ve Coello, 2013)

Şekil 1.1'de de görülen arama algoritması tahta üzerinde verilen pozisyona göre en optimum değeri veren hamleyi arar. Bu adımda kullanılan en yaygın algoritmalar olarak minimaks (Marsland, 1986), negamax (Campbell ve Marsland 1983), Alfa-Beta (α-β) (Knuth ve Moore, 1975), negascout (Reinefeld, 1983) ve quiescence (Beal, 1990) göze çarpmaktadır. İki oyuncuyla oynanan oyunlar için en kolay yaklaşımı oluşturan minimaks algoritmasının yanında ağaca ait bütün seviyelerde aynı operatörü kullandığı için programlaması daha kolay olan negamax algoritması da ağaçtaki en iyi değeri aramak için en çok kullanılan algoritmaların başında gelmektedir. İki oyuncuya ait oyunlarda temel teşkil eden yöntem ise α-β algoritmasıdır. α-β, algoritması gereksiz bazı düğümleri eleme avantajına sahip olmakla beraber bu yöntemin bir türevi olan

negascout da bulunmaktadır. Quiescence algoritması ise taş değişimleri gibi durumları stoklamak için arama ağacını genişletmesi ile bilinmektedir (Vázquez-Fernández ve Coello, 2013).

Arama algoritması ile birlikte satranç motorunun en önemli aşamasını oluşturan amaç fonksiyonu ise satranç pozisyonlarının bilgilerini depolayan bir dizi ağırlıklarla ifade edilen fonksiyondur. Bu ağırlıkların başarılı bir şekilde uyarlanması satranç motorunun oyunu daha iyi oynayabilmesini sağlamaktadır. Hamle üreteci ise, tahta üzerinde verilen pozisyona göre tüm olası hamleleri hesaplayabilen bir yapıdır (Vázquez-Fernández ve Coello, 2013).

Günümüzde neredeyse tüm satranç motorlarının açılış ve oyun sonu veritabanları bulunmaktadır. Böylece standart açılışlar sayesinde başlangıç hızlı olabilmekte ve 6 taşa kadar oyun sonları veritabanları sayesinde oyun kazanılabilmektedir. 7 taşa kadarki oyun sonları ise yüksek boyutu gereği çok fazla kullanılmamaktadır.

Stratejik oyunlar için birçok çözüm öne sürülmekle birlikte işbirlikçi ve işbirlikçi olmayan oyunların çözümü için John Von Neumann Oyun Teorisi'ni ortaya atmış ve Minimaks teoremini bulmuştur. Daha sonra yine John Forbes Nash bu teoriyi genelleştirerek denge teoremini ortaya koymuştur. Bunun dışında oyunların doğrusal programlama ile (Simplex yöntemi vb.) çözümleri de mevcuttur (Jianhua, 1988). Literatürde sezgisel yöntem kullanan birçok çalışma Minimaks teoremini kullanmaktadır. Sezgisel yöntemlerin dışında önceden oynanmış oyunları baz alarak karar veren sınıflandırıcıya dayalı yöntemlerde bulunmaktadır.

İlk satranç programı oluşturulduğundan bu yana çok sayıda yeni yöntem ve algoritmalar programa dâhil edilmeye çalışılmıştır. Satranç programları genel yapıları itibariyle amaç fonksiyonu ve arama mekanizması olarak iki ana yöntemden oluşurlar. Amaç fonksiyonu, herhangi bir hamlenin değerini sayısallaştırarak daha iyi hamlelerin tespitini kolaylaştırırken, arama

mekanizması ise belli bir derinlikte yapılabilecek tüm hamleleri bulmada kullanılır.

Literatürde yapılan çalışmalar incelendiğinde, daha derine inmek yerine küçük derinlikte olası tüm hamleler arasındaki en iyi hamleyi bulmaya odaklanılmıştır. Bunun için amaç fonksiyonları kullanılmakta olup, taşların değerleri, tehdit edilme durumları, yapabilecekleri hamle sayısı, tahta üzerindeki konumları vb. özellikleri kullanılarak o hamlenin değeri elde edilebilmektedir. Dolayısıyla literatürdeki çoğu yöntem belli bir derinlikteki tüm olası hamleleri hızlı bir şekilde α-β ve Negascout gibi popüler arama yöntemleri ile taradıktan sonra her bir pozisyonun değerini amaç fonksiyonları ile belirlerler. Yapılan araştırmalar sonucunda satranç oyununda sezgisel yöntemlerin çoğunlukla amaç fonksiyonu için kullanıldıkları görülmektedir. Özellikle genetik algoritmanın daha sık kullanıldığı göze çarpmakla birlikte, sezgisel yöntemler, amaç fonksiyonunun parametrelerini iyileştirmede kullanılmaktadır. Böylece çeşitli kabul görmüş bilgilere dayanarak en iyi hamleler tespit edilmeye çalışılmaktadır. Fakat, budama yapılmasına rağmen belli bir derinlikteki olası hamleleri arama işlemi çözülememiş bir problem olmayı sürdürmektedir.

Bu çalışmada çözülememiş bu problem üzerinde durulmuş olup, uzayın tümünün aranması yerine çözüm vadeden bölgelerde sınırlı sayıda arama yapılması sağlanmıştır. Bu sayede daha derine inilerek, uzun vadede daha iyi sonuca götürebilecek hamlelerin bulunma olanağı sağlanmıştır. Geliştirilen yeni yöntemle tüm olası hamleler araştırılmadığı için her zaman en optimum sonuca ulaşılamayabilir. Fakat, diğer hiçbir satranç programının uygulamadığı bu arama yöntemi ile çok daha ileriki hamlelerin tespit edilebilme olasılığı ortaya konmuştur. Bu durum satranç ustalarının bile oyunun gidişatını bu kadar etkileyecek bir hamleyi öngörememesinin yanı sıra, mevcut satranç programlarının da bu kadar sonraki hamleyi hesaplayamamaları sebebiyle geliştirilen yeni yöntem hem insan oyuncular arasında hem de bilgisayar programları arasında büyük bir avantaja sahiptir.

Bu çalışmada amaç fonksiyonunu oluşturan öznitelikler, 34 adet ağırlık değeri ile oluşturulmuştur. Ayrıca piyon, at, fil, kale ve vezirin değeri tehdit edildiklerinde yarıya inmektedir. Yöntemin başarısı diğer yöntemlere göre umut vaat edici bir seviyededir. Arama yöntemi olarak ise kümeleme tabanlı global optimizasyon (KTGO), yapay arı kolonisi (YAK), parçacık sürü optimizasyonu (PSO), geliştirilmiş yarasa algoritması (GYA) ve genetik algoritma (GA) kullanılmıştır. Ayrıca geliştirilen yönteme ait deneysel çalışmalar sayesinde sezgisel algoritmaların birbirleriyle kıyaslanma bulguları da elde edilmiş olup, hangi yöntemin satrançta arama için daha elverişli olduğu tespit edilebilmiştir. Sezgisel algoritmaların birbirleriyle yaptıkları karşılaşmalara ek olarak bazı oyun sonu hamleleri tespit edilmeye çalışılmıştır. SCHACKNYTT (chessnews)'de yayınlanan ünlü satranç oyuncularının oynadığı popüler oyunlardaki stratejik hamlelerin tahmini de deneysel çalışma için kullanılmış olup bu problemler için düşünülmüş değer hesaplama yöntemlerine göre algoritmaların puanları hesaplanmıştır. Ayrıca yazılan, kullanıcı arabirimi olan program insana karşı ve 15 adet satranç motoruna karşıda oyun oynayabilmektedir.

2. KAYNAK ÖZETLERİ

Bu tez kapsamında gerçekleştirilen uygulama ve değerlendirme aşamalarında önemli ölçüde yararlanılan, bugüne kadar gerçekleştirilmiş araştırma çalışmaları aşağıda kronolojik sırayla ele alınarak kısa özetleri ve sonuçları değerlendirilmiştir.

Moriaarty ve Miikkulainen (1994), çalışmalarında Otello programının arama parametrelerini Yapay Sinir Ağları (YSA) ile eğitmişlerdir. Arama mekanizması üzerine geliştirdikleri yöntem kendilerinin de belirttiği üzere satranç gibi kompleks bir oyun için uygun değildir.

Cazenave (2001), çalışmasında Go oyununda arama ve kontrol kurallarının öğrenilmesi için bir yöntem geliştirmişlerdir. Fakat Go oyununun arama mekanizması satranç oyunundan farklı olup, yöntem satranç oyunu için uyarlanamaz bir yöntemdir.

Björnsson ve Marshland (2002), satranç oyununda arama uzantılarının otomatik uydurulabilmesi için bir yöntem geliştirmişlerdir. Doğru hamleleri içeren test pozisyonlarını ve bir dizi optimize edilmesi gereken parametreyi dik iniş metodu kullanarak uydurmuşlardır. Geliştirdikleri program tüm pozisyonları işleyerek ve her bir pozisyon için düğüm sayılarını çözüm bulunana kadar kaydetmektedir. Tüm pozisyonlar için toplam düğüm sayısını minimize etmeye çalışmışlardır. Arama için 4 parametre değerinin optimize edildiği çalışmaları çok sayıda parametre içeren arama mekanizmaları için etkin kullanılamaz. Arama stratejisine farklı bir bakış açısı getirilmiştir. Bu çalışma satranç oyununda arama stratejisine sezgisel yaklaşımların öncüsü olmuştur.

Hong vd. (2002), çalışmalarında GA tabanlı oyun ağacı araması gerçekleştirmişlerdir. Amaç fonksiyonu yerine arama stratejisi geliştirilmesi açısından az sayıda rastlanan çalışmalardan biri olmakla birlikte, aramaların genellikle yeteri kadar derinde yapılamadığını belirtip 1 oyunculu oyun ağacı baz alınarak, ağacın daha hızlı ve etkili aranmasını gerçekleştirmişlerdir.

Özellikle zaman kısıtlı oyunlarda bu yöntemin klasik derine arama algoritmasından daha iyi olduğunu belirtmişlerdir.

Fogel vd. (2004), evrimsel algoritma kullanarak kendine karşı oynadığı oyunlarla satranç oynamayı öğrenmiştir. Taşların pozisyonları, materyal ve pozisyon değerleri YSA da kullanılarak satranç tahtası konfigürasyonlarını öğrenmiştir. Eğitim sonrasında program ELO değerini 400 puan arttırmıştır. En iyi hamlenin seçimi için minimaks algoritması α-β ile budanarak kullanılmıştır. Uygun çalıştırma zamanı için derinlik 4 seçilmiştir. Pocket Fritz 2.0 satranç programı ile toplamda 12 maçtan oluşan maç serisinde 9 oyunu kazanırken 2 oyunda yenilirken, 1 oyunda berabere kalarak 2550 ELO değerine sahiptir.

Hauptman ve Sipper (2005), geliştirdikleri güçlü satranç oyun sonu sisteminin incelemesini yapmışlardır. Genetik programlamanın kullanıldığı yöntemlerini 4 oyundan oluşan karşılıklı maçla CRAFTY'e karşı test etmişlerdir. Performans değerlendirmesinde 2614 ELO derecesine sahip CRAFTY ile yaklaşık berabere kaldığını belirtmişlerdir.

Duro ve Oliveira (2008), çalışmalarında sonraki hamle seçimi için minimaks algoritması ile birlikte α-β budamanın kullanıldığı satranç programı geliştirmişlerdir. Bu programın amaç fonksiyonu taşların cinsi, sayısı, şah korunumu gibi özellikleri içermekte olup, ağırlıkları sezgisel olarak PSO ile bulunmuştur. Tavlama benzetimine göre daha hızlı öğrenme gerçekleştiren PSO hesaplama süresinde çok iyi olmasına rağmen başlangıç koşullarına göre çok hassas olmaktadır. Bu sebeple başlangıç koşullarını eğitimden önce bilinen bir yöntemden alınmasının faydalı olacağını belirtmişlerdir.

Bošković vd. (2010), çalışmalarında satranca ait amaç fonksiyonunu uyarlamak için diferansiyel gelişim algoritması (DGA) tabanlı bir yöntem kullanmışlardır. Amaç fonksiyonunun evrimsel algoritmalar kullanılarak uyarlanması sadece oyun sonu kazançlarını kullanan bir kara-kutu problemi düşünülebileceğini belirtmişlerdir. Bu yöntemde popülasyonlar birbirleriyle oyun oynarlar ve elde ettikleri iyi sonuçlara göre gelecek nesillerde hayatta kalırlar. Popülasyonlar

oyun oynamak için satranca ait amaç fonksiyonunu ve arama algoritmasını kullanırlar. Öznitelikleri ve çeşitli diğer teknikleri kullanarak öğrenme kapasitesine sahip programların daha önceden oluşturuldukları görülmektedir. Bunlar arasında Levinson ve Snyder'ın geliştirdiği MORPH satranç programının yanında Thrun'un (1995) geliştirdiği NeuroChess programı da oyuna ait sonuçları kullanarak öğrenme kapasitesine sahiptir. Uyarlama aşamasında kullandıkları DGA geçmiş mekanizması ve rakip tabanlı optimizasyona sahip olup her nesildeki popülasyonlar mutasyon, çaprazlama ve seçim süreçlerine tabi tutulmaktadır. Önerdikleri yöntem BBChess satranç programına ait amaç fonksiyonunu uyarlamakta başarısı olmuştur. Geçmiş mekanizmasının kullanıldığı ve kullanılmadığı iki durum karşılaştırıldığında, geçmiş mekanizmasının kullanıldığı sonuçlar 155,2 puan daha fazla çıkmıştır.

Paulsen ve Fürnkranz (2010), çalışmalarında mevcut veritabanlarını kullanarak destek karar makineleri ile amaç fonksiyonu parametrelerini eğitmişlerdir. Hedefledikleri başarıya ulaşamamalarında hamle sırasında çok fazla derine bakılmamasının sebep olduğunu öne sürmüşlerdir. Fakat amaç fonksiyonunda da yer alan taşların konumuna bağlı değerlerin bulunmasında başarılı olduklarını ifade etmişlerdir.

David – Tabibi vd. (2010), çalışmalarında amaç fonksiyonuna ait özniteliklerinin değerlerini eğitmek için GA kullanmışlardır. Yaklaşık 50 iterasyon üzerinden 35 saatte elde edilen parametreler kullanılarak satranç test problemleri üzerinde CRAFTY ve FALCON ile karşılaştırıldığında daha başarılı olduklarını belirtmişlerdir.

David – Tabibi vd. (2011), çalışmalarında bir global optimum noktası bulmak yerine parametrelerini ayarlayabilen bir amaç fonksiyonu uyarlamaya çalışmışlardır. Amaç fonksiyonuna ait parametrelerin otomatik uyarlanması için GA'dan yararlanmışlardır. Her bir nesilde oluşturdukları rastgele satranç pozisyonlarını hesaplamak için literatürde yer alan güçlü bir satranç motoru olan FALCON'u kullanmışlardır. Kendi amaç fonksiyonlarının parametrelerini uyarlarken FALCON'a ait parametreler yerine sadece üretilen parametre

sonucunu kullanmışlardır. Sonuç olarak geliştirdikleri amaç fonksiyonu kullandıkları uzman program FALCON'a göre daha az sayıda parametreye sahip olup çok daha hızlıdır. Satrançta piyonun değerinin 1/100'üne centipawn denilmekle birlikte satranç programlarında hesaplama birimi olarak kullanılmaktadır. David-Tabibi ve arkadaşları GA'da 300 nesil kullanmış olup 35. nesilde amaç fonksiyonlarının değeri 50 centipawn'ın altına inmiştir. Bu aşamada satranç taşlarının cinsleri gibi yüksek parametre değerleri iyi bir şekilde uyarlanmış olup diğer küçük değerli parametrelerin tam olarak uyarlanması ise 300 neslin sonunda gerçekleşmiştir. Son nesilde ortalama hata 28 centipawn olup bu pozisyon FALCON'da hesaplandığında hatanın 30 centipawn olduğunu tespit etmişlerdir. Yaklaşık 8 dakikada hesapladıkları 300 nesil'e ait değerler ile uzmanın ürettiği değerlerin yaklaşık aynı olduğunu vurgulamışlardır. Geliştirilen algoritma (Evol*) ile FALCON'u karşılaştırmak için her biri 10 dakikadan oluşan 1000 adet oyun oynanmış olup yeni yöntemin bir parça daha iyi olduğu görülmüştür. Evol*'ın diğer programlara karşı performansını incelemek için CRAFTY, JUNIOR, FRITZ ve HIARCS programlarına karşı 300 oyun üzerinde test etmişlerdir. Evol* ayrıca taktiksel gücü hesaplamak için 879 pozisyon içeren Encyclopedia of Chess Middlegames (ECM) test setinde denenmiştir. Her bir programa doğru hamleyi yapması için 5 saniye vermişler ve sonuçlarını listelemişlerdir. Normal ortalama bir bilgisayarla 2008 Dünya Bilgisayarlar Arası Satranç Şampiyonasına katılarak hız ile ilgili turnuvada 2., normal turnuvada ise 6. olmuşlardır. Bu kötü sonuçlara donanım handikabının sebep olduğunu vurgulamışlardır.

Al-Khateeb ve Kendall (2011), çalışmalarında herhangi bir öğrenme algoritmasının eğer oyun ağacında daha derinlere taranmasının imkanı olursa daha iyi sonuçlar alınacağını belirtmiş olup sezgiselliğin önemini vurgulamışlardır. Deneysel çalışmalarında daha derine aramanın yapılmasının dama oyununa ait programın performansını arttırdığını ve öğrenme yeteneğinde önemli etkileri olduğunu belirtmişlerdir. Evrimsel bir algoritma olan Blondie24'ün mimarisinde de gösterildiği gibi daha derinlerde aramanın başarı oranını artıracağını belirtmişlerdir. Ayrıca her derinlik seviyesine geçişin aynı etkide olmadığını, örneğin 3 derinlik yerine 4 derinlikte aramanın 2

derinlik yerine 3 derinlikte aramaya göre performans farkının daha fazla olduğunu, bu sebeple olabildiğince fazla derinde çözümün aranmasının daha iyi olacağını belirtmişlerdir.

Seralathan (2011), tezinde kompleks kararlar için birkaç farklı strateji üzerinde durmuştur. Bunlar; satranç oynama sistemine örnek Deep Blue, amaç fonksiyonunun eğitilmesi için popüler olarak kullanılan 3 algoritmanın yanı sıra, satranç oyunu için adaptif olmayan öğrenme stratejisi kullanan satranç oyun sistemidir. Bu 3 öğrenme algoritması en iyi hamleyi veren minimaks yöntemine dayanmaktadır. İki algoritmanın amaç fonksiyonunu geliştirmek için sinir ağları metodunu kullanırken, diğeri için GA metodunu kullanmıştır. Son olarak çalışmasında, popülasyon dinamiklerini değiştirerek etkilerini ve çözüm uzayında aramanın nasıl yapıldığını analiz etmiş, satranç oyunu için önemli hamleler oyuncunun kromozomlarına yeni parametre olarak eklemiştir. GA'nın satranç oyunu gibi öngörü gerektiren problemlerde henüz yeterli olmasa da, diğer öğrenme tekniklerinden daha esnek olduğunu ve bu sebeple gelecek çalışmalarda geliştirilip daha iyi sonuç verebileceğini belirtmiştir.

Iqbal vd. (2012), çalışmalarında var olan estetik hesaplama modelini üç hamlede mat problemleri için geliştirip, uygun oyun sonu problemlerine uygulamışlardır. Çalışmalarının estetik ve güzellik algılaması bakımından daha uzun ve sofistike olduğunu, bu nedenle oyunda güzelliği değerlendirmek için makine kapasitesi açısından iyi bir test olduğunu belirtmişlerdir. Geçerli doğrulama yöntemlerine dayalı bu gibi estetik model için deneysel çalışmaların başarılı sonuçlar verdiği anlaşılmıştır. İlk deneyde, yeni modelin etkin olduğu program bileşik çalışmaları ve gerçek oyunlardan alınmış pozisyon dizilimlerini doğru bir şekilde ayırmıştır. İkinci deneyde ise, estetik hesaplama değerlendirmesinin pozitif korelasyon gösterdiği ve insan estetik uzman değerlendirmesinden daha iyi olduğu gözlenmiştir. Bu model sayesinde satranç hesaplamasının sınırlarının zorlanacağını ayrıca oyuncu ve hakemlere faydalı olacağını, geleneksel makine öğrenme yöntemlerinden daha etkili olduğunu ifade etmişlerdir.

Arenz (2012), tez çalışmasında satranç motorları üzerinde Monte Carlo ağaç arama yöntemini test etmiştir. Yapılan testler sonucunda yaklaşık 864 ELO derecesine ulaşmıştır. Yöntemin, Go oyunu için iyi sonuçlar verse de satranç oyunu için iyi sonuç vermediğini, minimaks ya da diğer arama yöntemlerinde daha kötü sonuçlar verdiğini belirtmiştir.

Vázquez-Fernández vd. (2012), çalışmalarında ön öğreticisiz YSA kullanarak satranç taşlarına ait pozisyon değerlerini elde etmişlerdir. YSA'ya ait ağırlıkların hesaplaması sonrasında geliştirdikleri satranç motorunun ELO seviyesi 1745'ten 2178'e çıkmıştır. Kullandıkları amaç fonksiyonu taşın kendi değeri ve pozisyon değerinden oluşmaktadır. Taşlara ait materyal değerleri sabit tutulup bu değerler piyon için 100, at ve fil için 300, kale için 500 ve vezir için 900'dür. Bu değerler satranç taşlarına ait teorik değerler ile örtüşmektedir. Taşlara ait pozisyon değerleri ise dinamik tutulup pozisyon, merkezi kontrol etme, hareket kabiliyeti gibi birçok etmene bağlı olarak değişmektedir.

Iba (2012), çalışmasında paralel genetik algoritma kullanarak şah-kale-şah (ŞKŞ) oyun sonu probleminin sınıflandırma başarısını arttırmaya çalışmıştır. Çoğu sınıflandırma problemine başarılı sonuçlar veren klasik genetik algoritmanın ŞKŞ probleminde yetersiz kaldığını öne sürüp, genetik algoritmanın performansını arttırmaya çalışmıştır. Bu sebeple paralel genetik algoritmayı (Iba vd., 2008) kullanarak daha etkili sonuçlar elde etmiştir. Deneysel çalışmada eğitim için toplamda 28000 olan örnek arasından rastgele seçilen 1000 tanesi kullanılmıştır. Bu çalışmada ayrıca bazı satranç problemleri YSA kullanılarak sınıflandırılıp başarı oranları ölçülmüştür. Bu problemler arasında şah, kale - şah, piyon problemi ile şah, kale – şah problemleri yer almakta olup veri setleri UCI Machine Learning Repository (Bache ve Lichman, 2013) de yer alan Chess veritabanından alınmıştır. Tüm problemler için sınıflandırıcı yöntem olarak YSA kullanılıp ağın yapısında girişte 100, gizli katmanda 50 ve çıkışta 2 nöron bulunmaktadır. Veritabanındaki verilerin yarısı eğitim için diğer yarısı ise test için kullanılmış olup 1000 iterasyon sonucunda eğitim setindeki başarı oranı ölçülmüştür. Şah, kale - şah, piyon problemi, satranç oyununun sonunda beyaz oyuncuda geriye şah ve kalenin kaldığı, siyah

oyuncuda ise şah ve bir piyonun kaldığı durumlarda beyaz oyuncunun oyunu kazanıp kazanamayacağını ifade eder. Hamle sırasının beyaz oyuncuda bulunduğu problem 3196 örnekten oluşup her bir örnek 36 özniteliğe sahiptir. Bu öznitelikler satranç tahtasının durumunu ifade etmektedir. YSA ile sınıflandırma sonucunda eğitim setindeki başarı oranı %100 bulunurken test setindeki başarı oranı %98.93 olarak elde edilmiştir. Şah, kale – şah probleminde ise beyaz oyuncunun oyunu kaç hamle sonra kazanacağını ya da berabere bitireceği yer almaktadır. Hamle sırasının siyah oyuncuda bulunduğu problem 28056 örnekten oluşmakta olup her bir örnek 6 özniteliğe sahiptir. Bu öznitelikler beyaz şahın satır ve sütunu (bulunduğu konum), beyaz kalenin satır ve sütunu ile siyah şahın satır ve sütunudur. YSA ile sınıflandırma sonucunda eğitim ve test setindeki başarı oranı %49 olarak bulunmuştur. Elde edilen başarı oranları satranç oyununda bitime son hamle kalması durumunda iyi sonuçlar gösterirken birden fazla adım kaldığı durumlarda ise çok etkili olamamıştır.

Vázquez-Fernández ve Coello (2013), çalışmalarında satranç motoruna ait ağırlıkların uyarlanabilmesi için taktiksel ve konumsal satranç problemlerine dayalı uyarlanabilir bir evrimsel algoritma geliştirmişlerdir. Algoritmanın seçim mekanizması uygun hamleleri seçerken, popülasyon elemanlarından oluşan sanal oyuncuların veritabanında yer alıp doğru bir şekilde çözdükleri satranç problem sayılarını göz önüne alır. Yöntemleri sayesinde kendilerine ait satranç motorlarını 1760 ELO seviyesinden 2317 ELO seviyesine çıkarmışlardır. Seçim ve mutasyona dayalı eğitim mekanizmaları ile her iterasyonda sanal oyuncular birbirlerinden bağımsız olarak verilen problemi çözmeye çalışırlar. En iyi çözüme ulaşan bireyler seçilip kullanılır. 4 derinlik seviyesine göre gerçekleştirdikleri deneysel çalışmaları sonucunda en uygun sanal oyuncu sayısının 22 olduğunu tespit etmişlerdir. Bunun yanında taşlara ait normal taş değerlerinin satranç teorisindeki değerlerle yaklaşık aynı çıktığını bulmuşlardır. Elde ettikleri bazı taş değerleri at için 297.29, kale için 493.57 ve vezir için 907.25'tir.

David vd. (2014), çalışmalarında satranç programı için üst düzey usta seviyesinde bir amaç fonksiyonu ve arama mekanizmasını genetik algoritma kullanarak geliştirmişlerdir. Başlangıç parametreleri rastgele oluşturulan program, evrim aşamasını kullanıcıların Grandmaster oyunlarını içeren veritabanına göre yapmaktadır. Arama mekanizma yapısının çoğunlukla amaç fonksiyonuna ait bilgilerin altında yattığını ileri sürmüşlerdir. Dolayısıyla arama mekanizmasında da GA kullanmışlardır. Amaç fonksiyonları diğer bilinen programlardaki pozisyon, merkezi kontrol etme, hareket kabiliyeti vb. parametreleri de içeren ve GA tarafından uydurulan 35 adet parametreden oluşmaktadır. Deneysel çalışmalarını Dünya Bilgisayar Satranç Şampiyonası'nı daha önceden 2 kez kazanmış olan CRAFTY programına göre test etmişlerdir. Çalışmalarında ECM pozisyonlarını bulma sayılarını 654'e çıkarmışlardır. Popüler seçmeli arama yöntemlerinden olan çoklu kesilmiş budama, boş hamle budaması metotlarının kritik parametrelerini gereksiz derinlik gibi parametrelerini GA ile bulmuşlardır. Bu sayede arama yöntemleri ve budama işlemine ait parametreler otomatik belirlenmiş olup arama stratejisine farklı bir bakış açısı getirilmiştir.

Wirth ve Fürnkranz (2015), çalışmalarında satranç oyununda amaç fonksiyonunun otomatik oluşturulması için oynanan hamleleri içeren notasyonları kullanan bir sistem üzerine çalışmışlardır. Yüksek boyutlu veri setlerinde çalışma imkânı oluşturabilen bu yöntemin diğer klasik öğrenme algoritmaları ile birlikte kullanılmasının daha verimli olduğunu belirtmişlerdir. Bu oyun notasyonlarının taşınabilir oyun notasyonları olan pgn uzantılı dosyalarda yer aldığını ve Chessbase gibi içerisinde milyonları bulan oyunların yer aldığı veritabanlarından elde edilebileceğini belirtmişlerdir. Bu sayede kompleks amaç fonksiyonlarının da öğrenilebileceğini deneysel çalışmalarında göstermişlerdir.

Vuskovic (2015), çalışmasında minimaks yönteminden yararlanan paralelleştirilmiş bir satranç oyun sistemi oluşturmuşlardır. Bu oyun paralel yapısı için sadece umut vadeden hamleleri kullanmaktadır. Geliştirdikleri 8 ile

64 arasında işlemci kullanabilen sistem olan Achilles'in ELO değerinin 3100'ün üzerinde olduğunu belirtmişlerdir.

Stoiljkovikj vd. (2015), satranç taktik problemlerinin çözümünün insanlar için ne kadar derecede zor olduğunu tespit etmeye çalışmışlardır. Sezgisel arama ile bilgisayar tarafından çözülen problemler insanın belli bir alanda arama yapabilme kabiliyeti göz önünde tutularak değerlendirilmiştir. Zorluk derecesi olarak kolay, orta ve zor 3 sınıftan yararlanmışlardır. Yöntemleri anlamlı ağaçlara dayanıp pozisyonları zorluk sınıf etiketlerine göre sınıflandırmışlardır. Deneysel çalışmalarında 3 sınıftan oluşan zorluk tahmini için sınıflandırma işleminde yöntemlerinin %80 civarında başarı gösterdiğini belirtmişlerdir. Bu sayede bir satranç probleminin bir insana zor gelip gelmeyeceğini tespit edebilmişlerdir.

Yokoyama vd. (2015), asenkron paralel oyun ağacı arama yöntemi gücünü birbirine bağlı olan bilgisayarın sayısına bağlı olduğunu belirtmişlerdir. Oyun pozisyonunun paralelleştirilmesinde ana program, oyun ağacında yer alıp dağınık pozisyonlarda yer alan işçi olarak ifade edilen bilgisayarları kontrol eder. Yöntem parametreden bağımsız olması ve kısa süreli maçların hesabında kullanılabilmesi önemli avantajları arasında sayılabilmektedir. Yöntemin güçlü bir şekilde oynaması için 60 işçi ile birlikte kullanılması gerektiğini belirtip kazanma oranının mevcut yöntemler ile karşılaştırılabilir olduğunu ifade etmişlerdir. Yöntemlerini işçi program olarak adapte edebilmek için yüksek seviye satranç programlarından olan açık kaynak kodlu Stockfish programını kullanmışlardır.

Zakharov vd. (2016), çalışmalarında 7 taşlı satranç oyun sonu problemlerinin çözümünü içeren tablonun boyutunu blok sıkıştırma algoritması ile küçültmüşlerdir. Satranç oyun sonu çözümlerini içeren tablolar herhangi ekstra bir bilgiye ihtiyaç duyulmadan oyunu mata götürecek hamlelerin listesini içermekle birlikte, 7 taş içeren tablonun sıkıştırılmış hali bile 100 Terabaytın üzerindedir. Sıkıştırılmamış hali ise 1 Petabaytın üzerindedir. Çalışmaları

sonucunda blok büyüklüğünü 1 kilobayta düşürüp, bu sayede performans sağlayıp dosya boyutunu azaltmışlardır.

Anh vd. (2016), çalışmalarında insan ile fiziksel satranç oynayabilen bir robot sistemi geliştirmişlerdir. Görüntü kodları ile satranç hamlelerini tanımak için Visual Studio'dan yararlanmışlardır. Ayrıca en iyi hamleyi tespit edebilmek için bir satranç programı ve bunu gerçek bir satranç tahtasında oynatabilmek için de robot kolundan yararlanmışlardır.

Literatür çalışmalarında da görüldüğü gibi sezgisel yöntemler genellikle amaç fonksiyonunun parametrelerinin eğitilmesi için kullanılmış olup, en iyi hamlenin bulunması ile ilgili arama mekanizması örneklerine az sayıda rastlanmıştır. Son birkaç yılda yapılan çalışmalar incelendiğinde bu tez çalışmasının literatürdeki boşluğu dolduracağı düşünülmektedir.

3. MATERYAL VE YÖNTEM

Bu çalışmada sezgisel algoritmaların yanı sıra çeşitli satranç bilgileri ve kurallarından da yararlanılmıştır.

3.1. ELO Derece Sistemi

Arpad ELO tarafından geliştirilen ELO derecelendirme sistemi satranç oyununda oyuncuların yetenek seviyelerini ölçmek için kullanılan resmi sistemdir. 2009 Dünya Satranç Federasyonu (FIDE) verileri incelendiğinde 2200 ELO seviyesinin üzerinde 21079 kişi bulunurken, 2700 değerin üzerinde 32 oyuncu bulunmaktadır. 2800 seviyesinin üzerinde sadece 4 oyuncu bulunurken, yeni başlayan biri genellikle 1400 seviyenin altında olmaktadır (Paulsen ve Fürnkranz, 2010). TSCP'nin (Tom Kerrigan's Simple Chess Program) amaç fonksiyonuna ait parça ve pozisyon değerleri olan ELO derece puan ve seviyeleri Çizelge 3.1'de görülmektedir.

Çizelge 3.1. TSCP'nin amaç fonksiyonuna ait parça ve pozisyon değerleri (Fernandez ve Coello, 2013).

ELO Puanı	Seviye
2400 ve üzeri	Üst düzey usta
2200 – 2399	Ulusal usta
2000 – 2199	Uzman
1800 – 1999	A sınıf
1600 – 1799	B sınıf
1400 – 1599	C sınıf
1200 – 1399	D sınıf
1000 – 1199	E sınıf

3.2. Amaç Fonksiyonu

Satranç programı, tahta üzerindeki tüm olası hamlelerin değerini amaç fonksiyonu kullanarak belirler. Bu çalışmada satranç oyununda yeni bir arama yöntemi üzerinde durulmuş olup, amaç fonksiyonu olarak en iyi sonucu veren öznitelikler kullanılmıştır. Amaç fonksiyonunu oluşturan öznitelikler, Fernadez

ve arkadaşlarının çalışmalarında kullandıkları 33 adet ağırlığa ek olarak yine Fernandez ve arkadaşlarının başka bir çalışmalarında kullandıkları 1 adet ağırlık daha eklenip 34 adet ağırlık değeri ile oluşturulmuştur (Vázquez-Fernández vd., 2012; Vázquez-Fernández vd., 2013). Ayrıca Nabiyev'in kitabında kullandığı gibi ilk 5 öznitelikteki elemanların değeri tehdit edildiklerinde yarıya inmektedir (Nabiyev, 2012). Kullanılan amaç fonksiyonun denklemi Eşitlik 3.1'de görülmektedir.

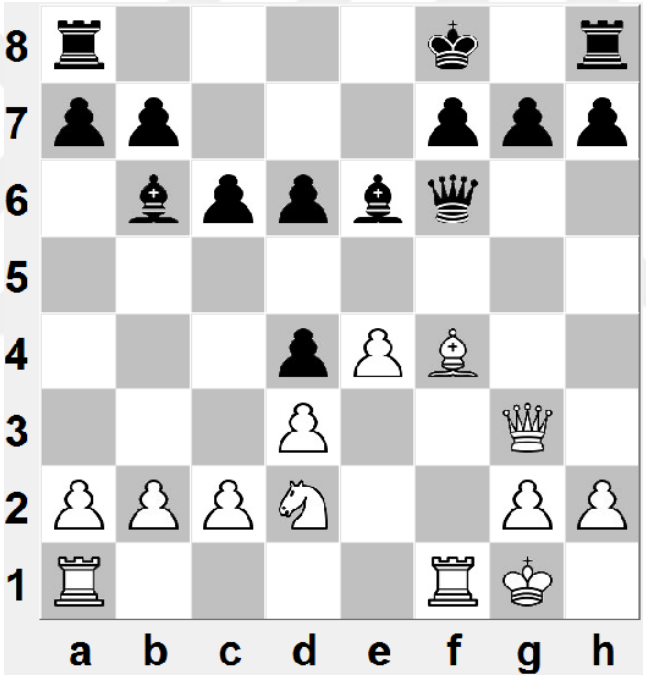
$$AmaçFonksiyonu = \sum_{j=1}^{34} Agırlık_j * f_j \quad (3.1)$$

Eşitlik 3.1'de verilen $AmaçFonksiyonu$, hamle sırası kendisinde olan oyuncuya ait toplam değeri ifade ederken, j değeri öznitelik sayısı, $Agırlık_j$ değerleri özniteliklerin ağırlıklarıdır ve Çizelge 3.1'de gösterilmiştir. Ayrıca f_j olarak belirtilen değişkenler ise amaç fonksiyonunun özniteliklerdir. Bunlar aşağıda listelenmiştir.

- $f_{piyon\_değeri}$ = Oyuncuya ait, tahta üzerindeki piyon sayısı.
- $f_{at\_değeri}$ = Oyuncuya ait, tahta üzerindeki at sayısı.
- $f_{fil\_değeri}$ = Oyuncuya ait, tahta üzerindeki fil sayısı.
- $f_{kale\_değeri}$ = Oyuncuya ait, tahta üzerindeki kale sayısı.
- $f_{vezir\_değeri}$ = Oyuncuya ait, tahta üzerindeki vezir sayısı.
- f_{duble_piyon} = Oyuncuya ait, tahta üzerindeki bir sütunda aynı renkten iki piyon olma sayısı.
- $f_{ayrık\_piyon}$ = Oyuncuya ait, tahta üzerindeki piyonun çevresinde aynı renkten başka piyon olmama sayısı.
- $f_{geri\_kalmış\_piyon}$ = Oyuncuya ait, tahta üzerindeki komşu sütunda aynı renk piyonun arkasında olma sayısı.
- $f_{geçer\_piyon}$ = Oyuncuya ait, tahta üzerindeki karşı tarafa ulaşabilmek için rakip piyonlar tarafından engellenemeyen piyon sayısı.
- f_{ortada_piyon}= Oyuncuya ait, tahta ortasındaki piyon sayısı (piyonların c4, c5, d4, d5, e4, e5, f4 ve f5 karelerindeki konumu).
- $f_{korunan\_at\_sayısı}$ = Oyuncuya ait, tahta üzerinde kendi piyonu tarafından korunan atların sayısı.

- $f_{at\_ileri\_hat}$ = Oyuncuya ait, tahta üzerinde ileri hatta olan at sayısı (Bulunduğu pozisyonda rakip oyuncunun piyonu tarafından tehdit edilmediğinde).
- $f_{at\_çerçeve\_0}$ = Oyuncuya ait, tahta üzerindeki a1-a8, b1-g1, h1-h8 ve b8-g8 arasındaki karelerin (dış çerçeve) üzerindeki atlarının sayısı.
- $f_{at\_çerçeve\_1}$ = Oyuncuya ait, tahta üzerindeki b2-b7, c2-f2, g2-g7 ve c7-f7 arasındaki karelerin üzerindeki atlarının sayısı.
- $f_{at\_çerçeve\_2}$ = Oyuncuya ait, tahta üzerindeki c3-c6, d3-e3, f3-f6 ve d6-e6 arasındaki karelerin üzerindeki atlarının sayısı.
- $f_{at\_çerçeve\_3}$ = Oyuncuya ait, tahta üzerindeki d4, e4, d5 ve e5 karelerinin üzerindeki atlarının sayısı.
- $f_{açık\_hat}$ = Oyuncuya ait, hiç piyonun yer almadığı sütunlardaki kalelerinin sayısı.
- $f_{kale\_yarı\_açık\_hat}$ = Oyuncuya ait, sadece rakip piyonların yer aldığı sütunlardaki kalelerinin sayısı.
- $f_{kale\_kapalı\_hat\_arkası}$ = Oyuncuya ait, her iki oyuncununda piyonlarının olduğu sütunda (kapalı hat), kendi piyonunun arkasındaki kale sayısı.
- $f_{yedinci\_satırdaki\_kale}$ = Oyuncuya ait, 7. satırdaki kalelerinin sayısı.
- $f_{at\_hamlesi}$ = Oyuncuya ait, atların yapabileceği hamle sayısı.
- $f_{fil\_hamlesi}$ = Oyuncuya ait, fillerin yapabileceği hamle sayısı.
- $f_{kale\_hamlesi}$ = Oyuncuya ait, kalelerin yapabileceği hamle sayısı.
- $f_{vezir\_hamlesi}$ = Oyuncuya ait, vezirlerin yapabileceği hamle sayısı.
- $f_{şah\_hamlesi}$ = Oyuncuya ait, şahın yapabileceği hamle sayısı.
- $f_{kale\_kapalı\_hat\_önü}$ = Oyuncuya ait, her iki oyuncununda piyonlarının olduğu sütunda (kapalı hat), kendi piyonunun önündeki kale sayısı.
- $f_{şah\_saldırı}$ = Oyuncuya ait, şahı ve etrafındaki pozisyonları tehdit edebilecek hamlelere sahip rakip taşların materyal değerleri toplamı.
- $f_{şah\_savunma}$ = Oyuncuya ait, şahı ve etrafındaki pozisyonları savunabilecek hamlelere sahip taşların materyal değerleri toplamı.
- f_{rok} = Oyuncuya ait şahın rok yapmış olma durumu (İkili değer alır, evet-hayır).

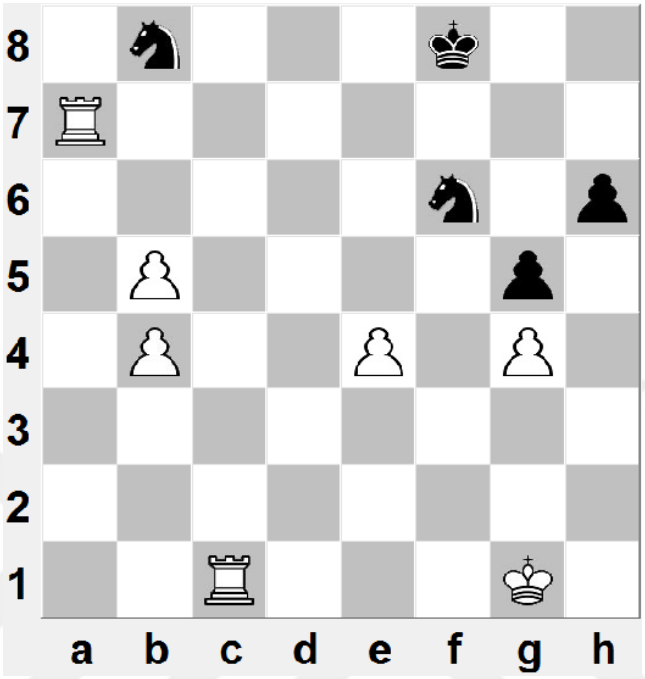
- $f_{şahın\_çevresindeki\_piyon}$ = Oyuncuya ait, şahın çevresindeki bitişik karedeki piyonların sayısı.
- $f_{öndeki\_fil}$= Oyuncuya ait, filin önünde olup filin hareketini engelleyen kendi piyonlarının sayısı.
- $f_{fil\_piyon\_hamlesi}$= Oyuncuya ait filin hareketini engelleyen piyonların gidebileceği hamle sayısı.
- $f_{çift\_fil}$ = Oyuncuya ait, iki adet fil olup olmaması durumu (İkili değer alır, evet-hayır).
- $f_{yedinci\_satırdaki\_çift\_kale}$ = Oyuncuya ait, 7. Satırda iki adet kale olup olmaması durumu (İkili değer alır, evet-hayır).



Şekil 3.1. Örnek satranç dizilimi 1

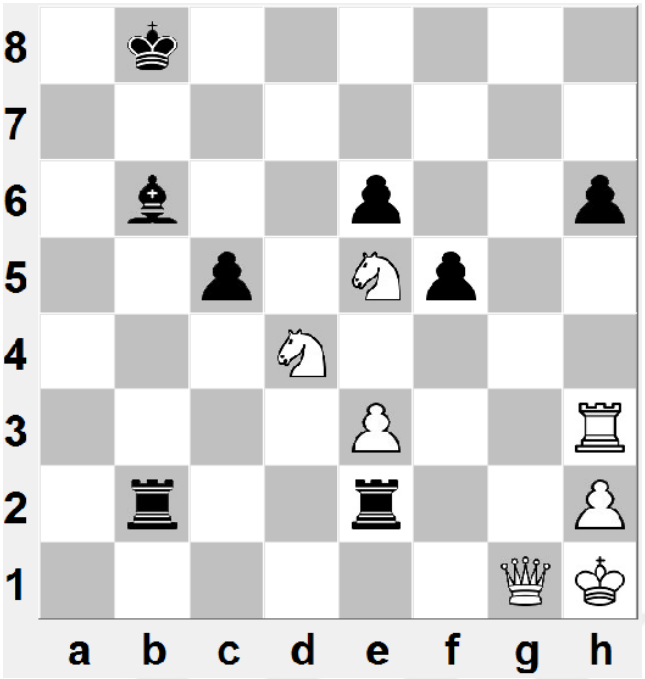
Şekil 3.1'de taş pozisyonları incelendiğinde beyaz oyuncu için amaç fonksiyonundaki özniteliklerden; $f_{piyon\_değeri} = 5$, $f_{at\_değeri} = f_{fil\_değeri} = f_{vezir\_değeri} = 1$, $f_{kale\_değeri} = 2$, e4'teki fil dolayısıyla $f_{ortada\_piyon} = 1$, d2'deki at dolayısıyla $f_{at\_çerçeve\_1} = 1$, $f_{kale\_yarı\_açık\_hat} = 1$, a1'deki kalede görüldüğü gibi $f_{kale\_kapalı\_hat\_arkası} = 1, f_{at\_hamlesi} = 4$, $f_{fil\_hamlesi} = 5$, a1'deki kale için $f_{kale\_hamlesi} = 4, f_{vezir\_hamlesi} = 10, f_{şah\_hamlesi} = 2$, f1'deki kale ve g3'teki vezir dolayısıyla $f_{şah\_savunma} = 1411$, f_{rok} = doğru, $f_{şahın\_çevresindeki\_piyon} = 2$ olduğu

görülmektedir. Eğer d4'teki siyah oyuncuya ait piyon olmasaydı $f_{şah\_saldırı} = 315.4$ olacaktı. Siyah oyuncu için amaç fonksiyonundaki özniteliklerden; $f_{öndeki\_fil} = 1$ ve $f_{çift\_fil} = doğru$ olduğu görülmektedir.



Şekil 3.2. Örnek satranç dizilimi 2

Şekil 3.2'de taş pozisyonları incelendiğinde beyaz oyuncu için amaç fonksiyonundaki özniteliklerden; b4 ve b5'teki piyonlar dolayısıyla $f_{duble\_piyon} = 1$, e4'teki piyon sebebiyle $f_{ayrık\_piyon} = 1$, b5 ve e4'teki piyonlar dolayısıyla $f_{geçer\_piyon} = 2$, c1 ve a7'deki kale $f_{açık\_hat} = 2$, a7'deki kale sebebiyle $f_{yedinci\_satırdaki\_kale} = 1$ olduğu görülmektedir. Siyah oyuncu için amaç fonksiyonundaki özniteliklerden; h5'teki piyon dolayısıyla $f_{geri\_kalmış\_piyon} = 1$, b8'deki at dolayısıyla $f_{at\_çerçeve\_0} = 1$, f6'daki at dolayısıyla $f_{at\_çerçeve\_2} = 1$ olduğu görülmektedir.



Şekil 3.3. Örnek satranç dizilimi 3

Şekil 3.3'de taş pozisyonları incelendiğinde beyaz oyuncu için amaç fonksiyonundaki özniteliklerden; e3'teki piyon dolayısıyla $f_{korunan\_at\_sayısı} = 1$, e5'deki at sebebiyle $f_{at\_ileri\_hat} = 1$, d4 ve e5'deki atlar dolayısıyla $f_{at\_çerçeve\_3} = 2$, h sütunundaki durum sebebiyle $f_{kale\_kapalı\_hat\_önü} = 1$ olduğu görülmektedir. Siyah oyuncu için amaç fonksiyonundaki özniteliklerden; c5'teki piyon dolayısıyla $f_{fil\_piyon\_hamlesi} = 2$, b2 ve e2'deki kale dolayısıyla $f_{yedinci\_satırdaki\_çift\_kale}$ = doğru olduğu görülmektedir.

Çizelge 3.2. Amaç fonksiyonundaki özniteliklere ait katsayılar

Sayı	Öznitelik	Ağırlık
1	Piyon_değeri	100.0
2	At_değeri	295.2
3	Fil_değeri	315.4
4	Kale_değeri	489.7
5	Vezir_değeri	921.3
6	Duble_piyon	-14.3
7	Ayrık_piyon	-23.2
8	Geri_kalmış_piyon	-19.5
9	Geçer_piyon	34.3
10	Ortada_piyon	19.7
11	Korunan_at_sayısı	16.2
12	At_ileri_hat	15.3

Çizelge 3.2. Amaç fonksiyonundaki özniteliklere ait katsayılar (Devam)

13	At_çerçeve_0	-10.3
14	At_çerçeve_1	14.1
15	At_çerçeve_2	19.7
16	At_çerçeve_3	26.4
17	Açık_hat	22.1
18	Kale_yarı_açık_hat	-5.7
19	Kale_kapalı_hat_arkası	-11.3
20	Yedinci_satırdaki_kale	49.6
21	At_hamlesi	27.8
22	Fil_hamlesi	34.1
23	Kale_hamlesi	37.2
24	Vezir_hamlesi	12.3
25	Şah_hamlesi	5.1
26	Kale_kapalı_hat_önü	9.3
27	Şah_saldırı	-67.8
28	Şah_savunma	52.3
29	Rok	51.2
30	Şahın_çevresindeki_piyon	49.3
31	Öndeki_fil	39.6
32	Fil_piyon_hamlesi	21.3
33	Çift_fil	27.0
34	Yedinci_satırdaki_çift_kale	50

3.3. Arama Mekanizması

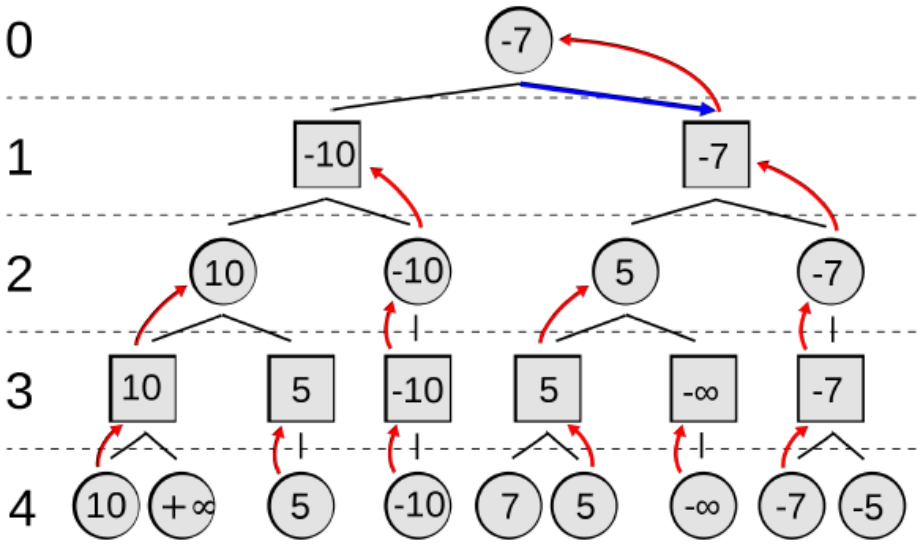
Satranç programlarında arama mekanizması belli bir derinlikte yapılabilecek hamleleri tespit etmeye yarar. Derinlik arttıkça olası hamlelerin sayısının inanılmaz boyutlara ulaşması sebebiyle literatürdeki diğer satranç programlarının popüler olarak kullandığı α-β ve Negascout gibi yöntemler sadece sınırlı bir alanı tarayabilmektedirler. Bu çalışmada bu soruna bir çözüm getirmek ve daha iyi bir satranç programı ortaya koyabilmek adına, arama mekanizmasında çeşitli sezgisel algoritmalar kullanılmıştır. Bu algoritmaların kendi özgünlüklerine göre oluşturdukları popülasyonlar daha önceki çalışmamızda belirttiğimiz hamle üretecindeki karşılıkları bulunarak ve daha sonra bu pozisyonların amaç fonksiyonu değerleri hesaplanarak yapılır. Popülasyonlarda üretilen değerle satranç oyunu ağaç yapısı olarak düşünüldüğünde dalları ve bunlara ait alt dalları ifade etmektedir. Böylece ağaç

üzerinde tüm alt dallar inceleneceğine belli bölgeler tespit edilir. Çalışmada hesap süresinin kabul edilebilir bir sürede çalışması için popülasyon ve iterasyon değerleri belli değerlerde tutulmuştur.

Arama mekanizmasında kullanılan sezgisel yöntemler olarak literatürde en çok kabul gören YAK, PSO, GYA ve GA algoritmalarının yanı sıra sezgisel yöntemler arasında daha hızlı çalışması ve lokal minimumları daha kesin tespit edebilmesi açısından ön plana çıkan KTGO da kullanılmıştır.

3.3.1. Minimaks yöntemi

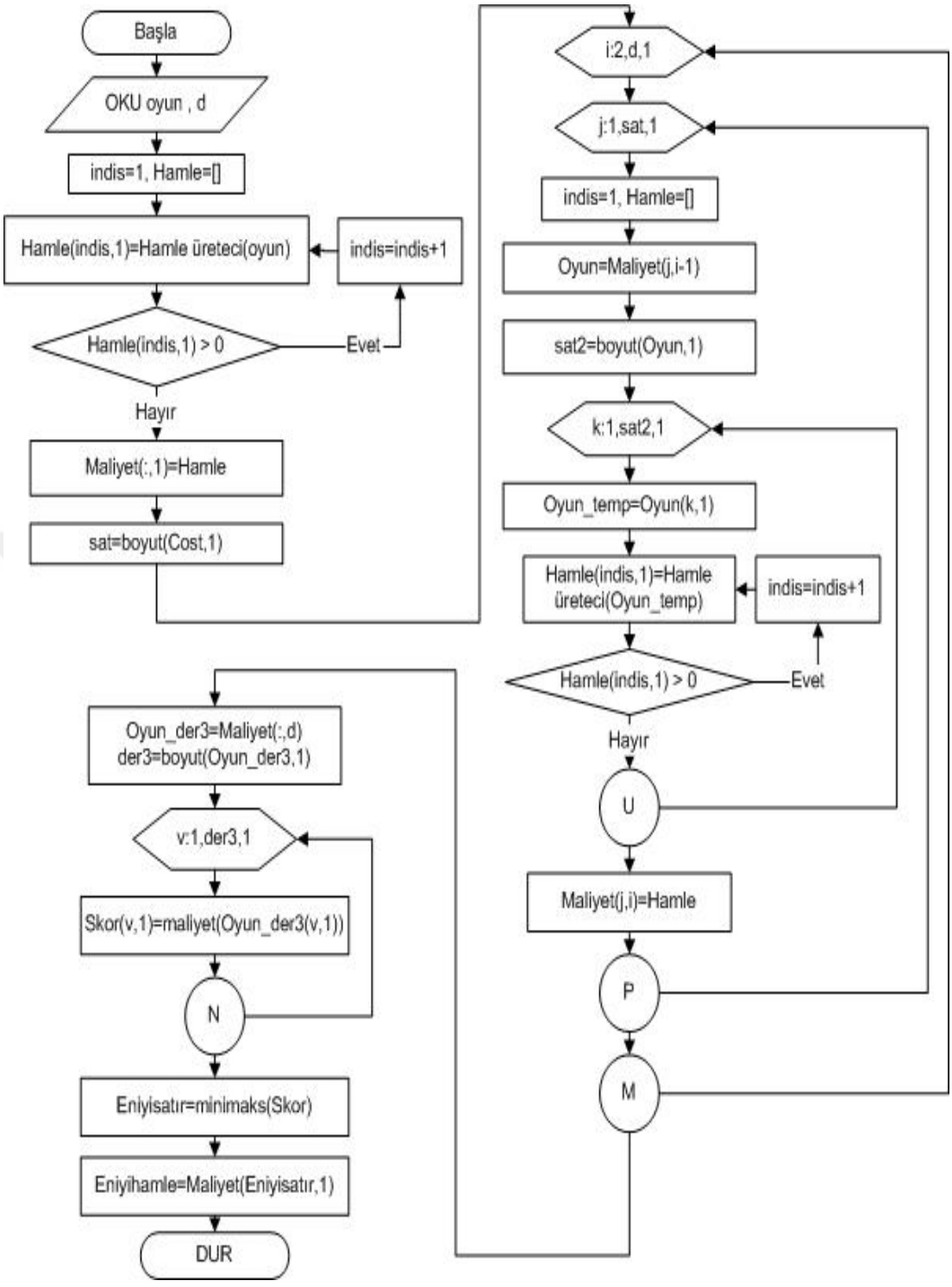
Oyunlarda bilgisayarların karar vermesine yardımcı olan ve satranç oyununa ait yapay zekânın temelini oluşturan minimaks teoremi 1928 yılında John von Neumann tarafından ortaya atılmıştır. Minimaks "Min" ve "Max" isimli iki adet oyuncuyu dikkate alıp Min isimli oyuncunun Max oyuncusunun alacağı miktarı minimum yapması olarak bilinir. Aynı şekilde Max oyuncusunun amacı da elde edeceği miktarın en yüksek olmasını sağlamaktır. Minimaks algoritmasının çalışma prensibinin en iyi anlaşılabileceği uygulama olarak ağaç arama algoritmaları gösterilebilir. Max ve Min'in tüm olası hamlelerinin bir ağacın dalları olduğu kabul edilirse Max ve Min oyuncuları sırasıyla ağaç üzerinde en yüksek ve en düşük değerleri amaçlar. Minimaks algoritmasına ait örnek bir gösterim Şekil 3.4'de gösterilmektedir. Minimaks yönteminde birçok optimizasyon yer almaktadır. Satranç gibi sıfır toplam oyunlarında bir oyuncunun herhangi bir kazancı rakibinin zararına olmaktadır. Bu tarz sıfır toplam oyunları her iki tarafın kazancının toplamının sıfır olduğu oyunlardır. Bu özellik $max\ (a, b) = -min(-a, -b)$ şeklinde özetlenebilir. Min ve Max'a ait aramaların özyinelemeli bir fonksiyona dönüştürülürmüş haline negamax denilmektedir (Von Neumann, 1928, Nabiyev, 2012).



Şekil 3.4. Minimaks algoritmasının Ağaç Arama algoritmasındaki kullanımı (Wikipedia, 2015)

Minimaks, amaç fonksiyonuna göre oluşturulan graf yapısında en uygun düğümü bulmada çok iyi olmasına rağmen satranç gibi derinliği arttıkça durum uzayı çok fazla büyüyen oyunlarda yetersiz kalabilmektedir. Bu sebeple GA, YAK vb. sezgisel yöntemlerin kullanılması da kaçınılmazdır.

Geliştirilen satranç oyununda arama yöntemlerinden minimaks'ın programın yapısına nasıl eklendiği akış diyagramlarıyla Şekil 3.5'de görülmektedir. Şekil 3.5'deki M, N, P ve U döngü sonunu göstermek için kullanılmış düğümlerdir.



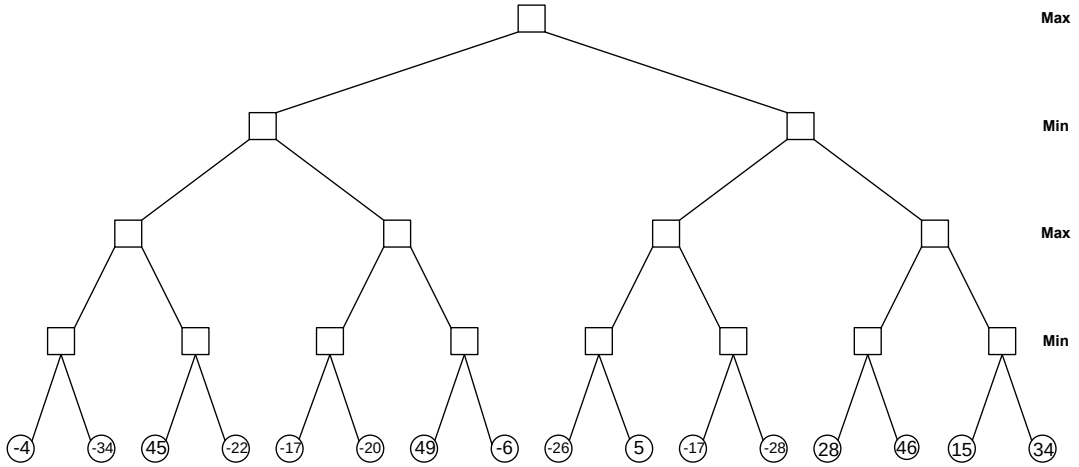
Şekil 3.5. Minimaks'ın satranç oyunundaki yapısının akış diyagramı

3.3.2. Alfa-Beta (α-β) budama

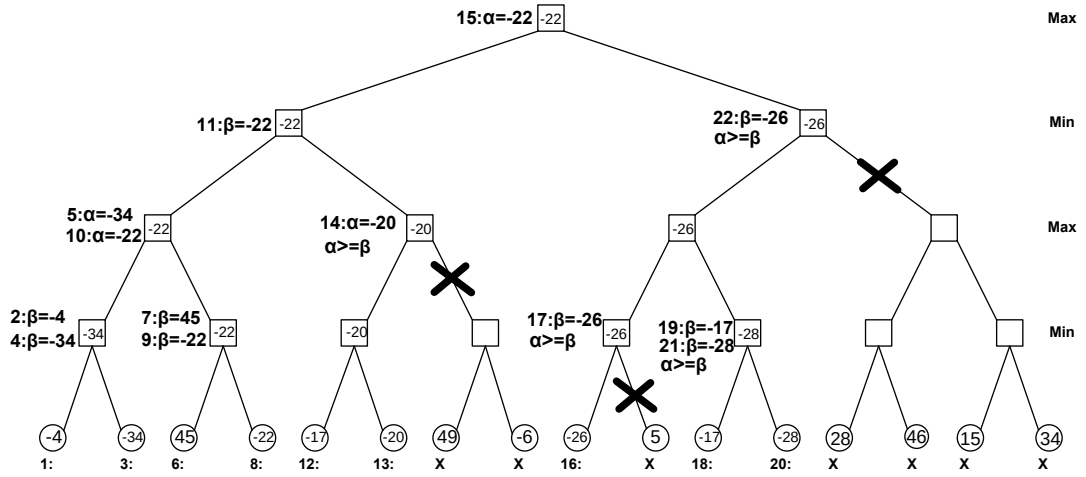
α-β budama minimaks algoritmasını hızlandırmaya yarayan bir yöntemdir. Minimaks algoritmasında çözüm ağacı oluşturulurken herhangi bir durum

değerlendirmesi yapılmamakta, tüm ağaç oluşturulduktan sonra elde edilen düğümler değerlendirilmektedir. Arama ağacındaki dal sayısı arttıkça değerlendirilecek durum ve en iyi yol seçimi için hesaplama zamanı kabul edilebilir seviyelerin üstüne çıkabilmektedir. Bu problemin önüne geçmek amacıyla 1956 yılında J. McCarty tarafından önerilen α-β budama ile arama ağacında birçok pozisyonun değerlendirilmeden elenmesi mümkündür. Ağaç oluşturulurken belirli sınırlamalar getirilmesi, minimaks algoritmasını daha etkin kılmaktadır (Knuth ve Moore, 1975; Nabiyev, 2012).

Bu yöntemin temel amacı kötü olmayan gidişin bulunmasıdır. Yöntem ismini α ve β olan iki değerden alır. Burada α, max oyuncusu için garantilenmiş en küçük değerlendirmeyken; β, max'ın alabileceği fonksiyon değerlerinin en büyüğüdür. Min oyuncusu için β garantilenmiş değerler arasında en kötüsüdür. Böylece aranan değer aralığı α-β aralığına indirgenmiş olur. Eğer herhangi bir durumda elde edilen değer bu aralık dışındaysa ve benzeri durumların değerlendirilmesi gereksiz olacağından ağacın belirli kısmı incelenmez. Yani budama işlemi gerçekleştirilmiş olur. Budama işlemi iki uç düğüm değerlendirildikten sonra gerçekleştirilmektedir. Budama işlemi aranacak düğüm sayısını azalttığı için istenilen sonuca ulaşılması minimaksa göre daha hızlı olmakla beraber bu oran derinlik sayısı arttıkça daha belirgin olmaktadır.



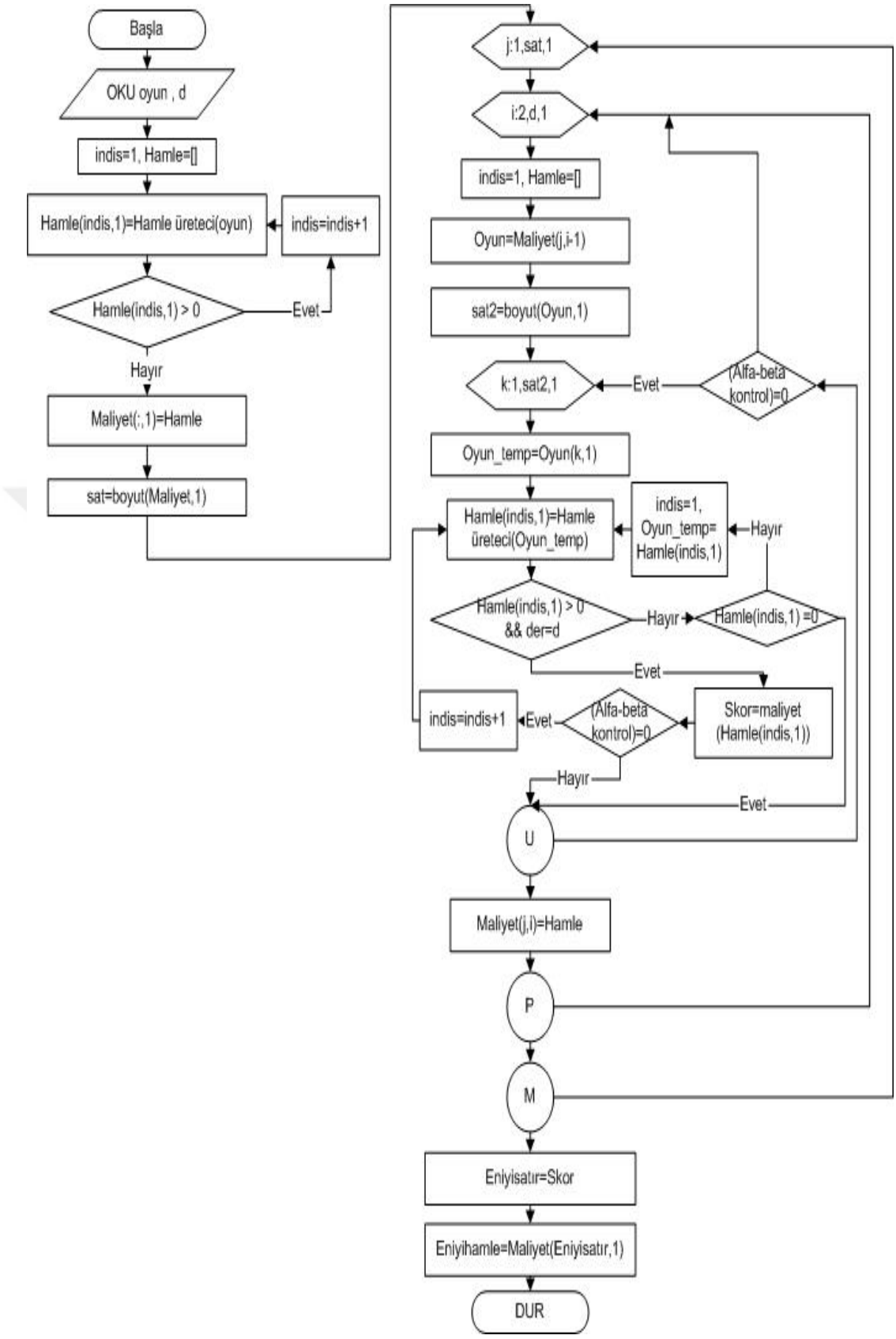
Şekil 3.6. Basit bir α-β problemi



Şekil 3.7. Basit bir α-β problemi çözümü

Şekil 3.6'da verilen problem için yapılacak olan α-β budama işleminde α o ana kadar Max düğümünde bulunmuş en büyük değer olurken, β ise Min düğümünde bulunmuş en küçük değer olmalıdır. Başlangıç değeri olarak α değeri -∞ iken, β değeri +∞ olarak belirlenir ve karşılaştırma işlemine başlanır. Eğer Max seviyesindeki bir durumun α değeri, hemen yukarısındaki durumun β değerinden büyükse diğer durumların incelenmesine gerek kalmaz ve budama gerçekleştirilir. Ayrıca Min seviyesindeki bir durumun β değeri hemen üstündeki durumun α değerinden küçükse diğer durumların incelenmesine gerek kalmaz ve budama gerçekleştirilir. Budama işlemi ilerledikçe α değeri artmakta, β değeri ise azalmaktadır. İşlem sonunda bu değerler eşitlenir. Yukarıdaki şekilde α-β budama işlemi uygulandığında en üstteki düğüm için -22 sonucu bulunmaktadır. Çünkü α değeri hiçbir zaman azalamaz hep artar, β değeri de hiçbir zaman artamaz hep azalır.

Geliştirilen satranç oyununda arama yöntemlerinden α-β budamanın programın yapısına nasıl eklendiği akış diyagramıyla Şekil 3.8'de görülmektedir. Şekil 3.8'deki M, P ve U döngü sonunu göstermek için kullanılmış düğümlerdir.



Şekil 3.8. α-β satranç oyunundaki yapısının akış diyagramı

3.3.3. Kümeleme tabanlı global optimizasyon yöntemi

Sezgisel yöntemler arasında daha hızlı çalışması ve lokal minimumları daha kesin tespit edebilmesi açısından ön plana çıkan KTGO kümeleme ve parabolik eğri uydurma yöntemlerine dayanmaktadır.

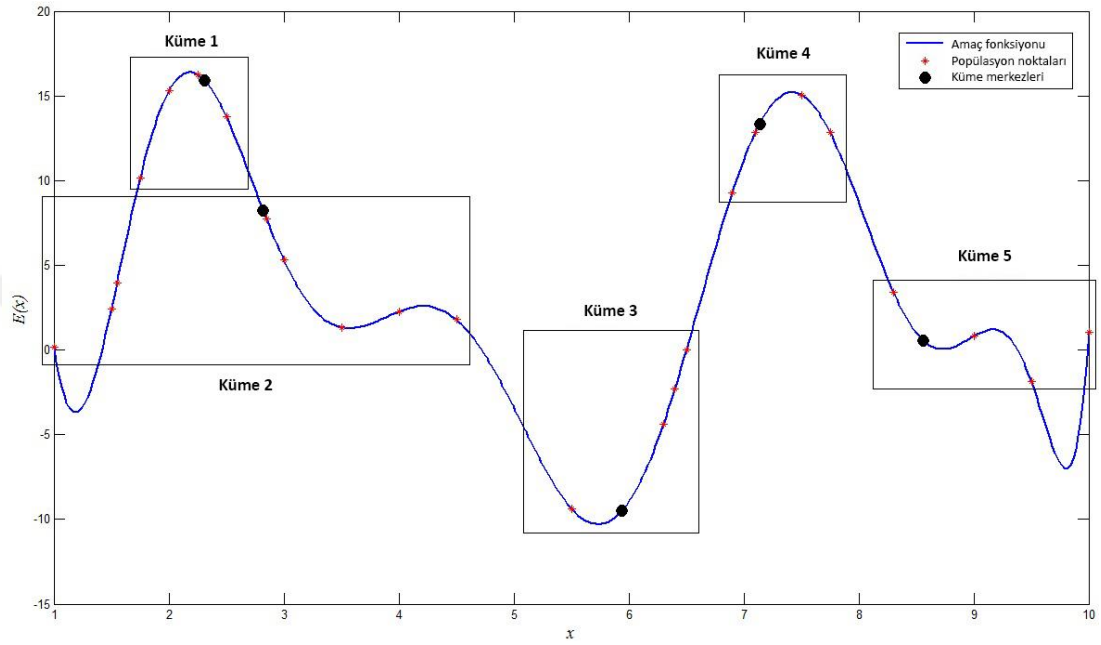
Bu yöntem genel olarak popülasyondaki elemanları ve bunlara ait performans değerlerini kullanarak kümeleme işlemi yapması, yeni popülasyonu bu küme merkezleri etrafından seçmesi ve kümelerin parabolik eğri uydurma ile minimum ya da maksimum noktalarının bulunması ön plana çıkmaktadır. Yöntemin basitleştirilmiş algoritması şu şekildedir (Pence vd., 2016):

Algoritma (KTGO)

1. Önceden belirlenen popülasyon büyüklüğüne göre ilk popülasyonu rastgele elemanlardan oluştur ve popülasyondaki her eleman için hata değerini hesapla.
2. Popülasyon-hata oranı ve önceden belirlenen küme sayısına göre Bulanık C-Ortalamalar kümeleme yöntemi ile kümeleme işlemi yap.
3. Fonksiyon değerlerine göre kümeleri sırala ve küme sayısının 1/3 oranında en iyi kümeleri seç.
4. Seçilen her bir kümeye ait elemanları kullanarak her boyut için ayrı ayrı parabolik eğri uydurma gerçekleştir ve oluşan eğrilerin iç bükey veya dış bükey olduğunu tespit et.
5. İç bükey eğrilerde minimum nokta hata değerinden daha küçük ise küme merkezini bu minimum nokta ile değiştir. Dış bükey eğriler için ise küme merkezini değiştirmeden kullan.
6. Önceki popülasyonun en iyi elemanları ile birlikte küme merkezi etrafından rastgele seçilen yeni elemanlarla yeni popülasyonu oluştur. Çıkarılan kümelere ait bireyler yerine belli aralıkta yeni noktalar ekle.
7. Yeni popülasyondaki elemanların değerlerini önceden belirlenmiş sınırlar içinde tut.
8. Yeni popülasyondaki her eleman için hata değeri hesapla.

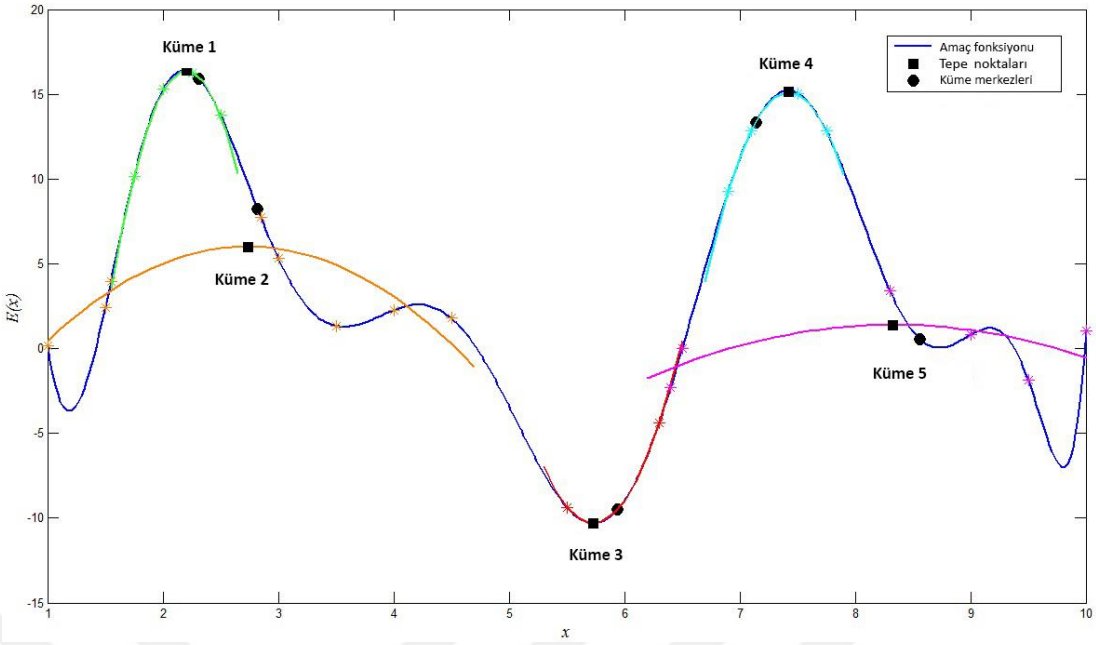
9. Hata değeri önceden belirlenen eşik değerinden küçükse veya hedeflenen tekrara erişilmişse algoritmayı sonlandır değilse 2. Adıma git.

Şekil 3.9'da oluşturulan popülasyonların kümeleme işlemi sonrasında küme merkezlerinin tespiti görülmektedir.



Şekil 3.9. Popülasyonların kümeleme işlemi ile küme merkezlerinin tespiti (Pence vd., 2016)

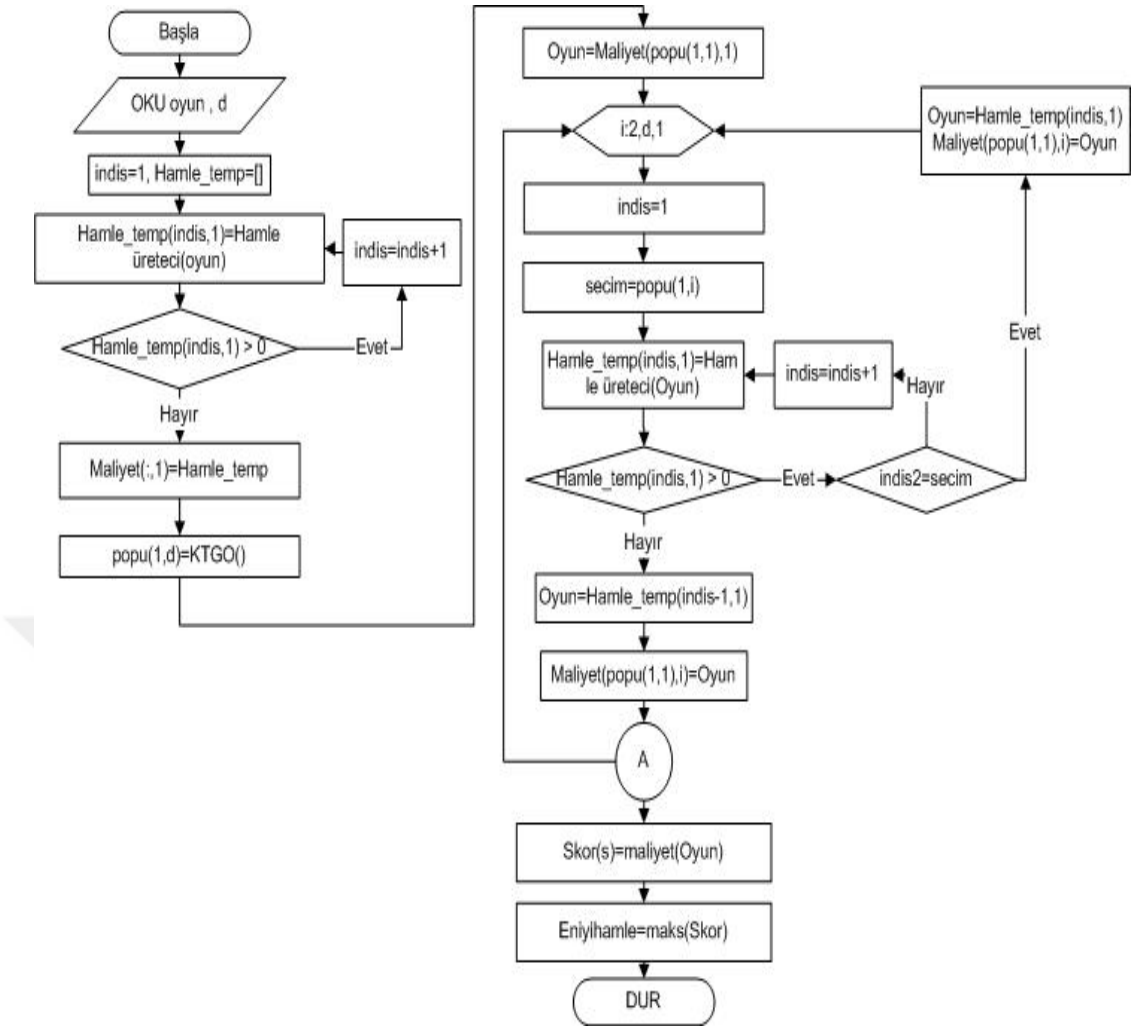
Şekil 3.10'da ise kümelere ait popülasyonlar kullanılarak parabolik eğri uydurma işlemi görülmektedir.



Şekil 3.10. Popülasyonların parabolik eğri uydurma ile minimum ve maksimum noktalarının tespiti (Pence vd., 2016)

Şekil 3.10'da görüldüğü gibi bölgesel parabolik eğrilerin tepe noktaları minimumları, maksimumları veya orta değerleri gösterir. Böylelikle global minimuma gidiş hızlandırılmaktadır.

Geliştirilen satranç oyununda arama yöntemlerinden KTGO'nun programın yapısına nasıl eklendiği akış diyagramlarıyla Şekil 3.11'de görülmektedir. Şekil 3.9'daki A döngü sonunu göstermek için kullanılmış düğümdür.



Şekil 3.11. KTGO'nun satranç oyunundaki yapısının akış diyagramı

Şekil 3.5, Şekil 3.8 ve Şekil 3.11'de görüldüğü gibi minimaks yöntemi gerekli gereksiz tüm uzayı tarayıp, en derindeki tüm düğümlerin de değerlerini hesaplamakta iken α-β yöntemi biraz daha fazla koşula sahip olmasına rağmen α ve β'nın kontrol edildiği yerlerde çoğu dalı pas geçebilmektedir. KTGO yöntemi ise diğerlerine nazaran sadeliği ile ön plana çıkmakla beraber popülasyonda verilen hamle numaralarını oluşturmaya çalışırken sadece en derindeki tek bir düğüm için değer hesaplamaktadır. Daha sonra her bir popülasyonun değerlerinden maksimum olanı (1.derinlikte maksimuma bakılır) en iyi hamle olarak belirlenir.

Ayrıca bu çalışmada KTGO'nun eksik yönlerinden biri olan, en iyi elemanların tutulduğu bellek mekanizmasının olmaması dezavantajı aşılmaya çalışılmıştır. Bunun için Tabu mekanizmasına benzer bir yapı kullanılmış olup, belli sayıda

iterasyon sonrasında elde edilen her bir ana daldan türetilen popülasyonların puanların ortalamasına göre en kötü ve en iyi olarak ikiye ayrılır. Daha sonra tespit edilen en kötü yarısındaki popülasyonların yerine, en iyi daldan türetilen rastgele elemanlar kullanılır. Böylece en iyi daldan türetilen elemanlardan oluşan umut vadeden bölgede daha çok popülasyon üretilmesi gerçekleştirilmiş olur. Bu sayede kötü popülasyonlar sistemden uzaklaştırılırken, daha çok iyi eleman oyun ağacına kazandırılır. Gücünü kümeleme yönteminden alan KTGO, bu yaklaşım ile gereksiz küme merkezleri ile de uğraşmayıp, sonuca daha iyi yaklaşabilmiştir. Süre açısından küçük bir dezavantaj olmasına rağmen başarı oranının artacağı açıktır.

3.3.4. Parçacık sürü optimizasyonu algoritması

PSO, 1995 yılında Kennedy ve Elberhart tarafından kuşların ve balıkların davranışları modellenerek geliştirilmiş sürü tabanlı sezgisel bir optimizasyon algoritmasıdır (Kennedy ve Eberhart, 1995). Parametre sayısının azlığı ve türev bilgisine ihtiyacı olmaması sebebiyle işletilmesi kolay bir yöntem olarak görülmektedir (Zhao vd., 2005; Juang vd., 2006).

PSO'da her bir birey parçacık olarak ifade edilir ve her bir parçacığın kendine ait pozisyon değeri, hız değeri, kendi optimum değeri ve sürüdeki optimum değeri vardır (Arumugam vd, 2008). Buradaki hız parçacığın yer değiştirme miktarını ifade etmektedir. Algoritmanın başında parçacığa ait ilk pozisyon ve hız değerleri rastgele olarak ya da açgözlü bir algoritma yardımıyla yapılır. Parçacığın her hareket sonunda yeni pozisyon değerleri algoritmanın başında verilen amaç fonksiyonuna göre uygunluk değeri hesaplanır (Zhang vd., 2007). Her iterasyonda parçacık, hızını ve konumunu iki değerle günceller. Bunlar; parçacığın o ana kadar bulduğu en iyi uygunluk değeri ve popülasyondaki parçacıklar tarafından elde edilmiş en iyi uygunluk değerine sahip olan çözümdür.

Sürünün hareketi istenilen durdurma kriterine kadar devam eder. Durdurma kriteri; algoritmanın başında belirlenmiş iterasyon sayısının tamamlanması,

minimum hata oranına ulaşılması, algoritma başında belirlenen optimum değere ulaşması ya da belirli iterasyon boyunca ya da belirli bir süre en iyi çözümün değişmemesi olabilir.

PSO'nun yakınsayamama sorununun önüne geçebilmek ve kararlı sonuçlar elde edebilmek için atalet ağırlığı kullanılmaktadır. Böylece mevcut hızın yeni hıza etkisi kontrol altına alınarak optimum değere ulaşma olasılığı artırılmıştır. (Shi ve Eberhart, 1998). Atalet ağırlığının büyük olması çözüm uzayında büyük aramalar yapılmasını sağlarken, küçük olması bölgesel aramalar yapılmasını sağlamaktadır (Eldem, 2014). Bu çalışmada da algoritmaya atalet ağırlığı eklenmiştir.

3.3.5. Geliştirilmiş yarasa algoritması

YA, Yang tarafından 2009 yılında önerilmiş sezgisel bir algoritmadır (Yang, 2010). Yarasaların karanlıkta yönlerini ses titreşimiyle yer tespiti (ekolokasyon) yaparak bulmalarından esinlenerek geliştirilmiştir. YA'nın, keşfetme ve keşfedilen çözümleri iyileştirme kabiliyetlerinin zayıf olmasından dolayı Yılmaz ve Küçüksille tarafından GYA geliştirilmiştir (Yılmaz ve Küçüksille, 2015).

$$v_i^t = v_i^{t-1} + (x_i^t - x_*) f_i \tag{3.2}$$

Eşitlik 3.2.'de v_i^t, *t.* anda *i.* bireyin hızını; x_i^t, *t.* anda *i.* bireyin konumunu; x_*, popülasyon içerisinde *t.* süreye kadar en iyi bireyin çözüm değerini; f_i, *i.* yarasaya ait frekans değerini temsil etmektedir.

YA'da, Eşitlik 3.2.'de verilen hız güncelleme eşitliğine yalnızca birinci terimin etki etmesi durumunda yakınsama hızı oldukça yavaşlayıp optimizasyon performansı zayıflarken, ikinci terimin eşitliğe etki ettiği durumda erken yakınsama problemiyle karşı karşıya kalınmaktadır. Yılmaz ve Küçüksille çalışmalarında hız ve konum bilgilerinin güncellenmesinin PSO algoritmasına benzerliğini belirterek, bu algoritmada hız vektörüne uygulanan atalet katsayısını YA'na uygulamışlardır (Yılmaz ve Küçüksille, 2015).

Atalet ağırlığı, başlarda bireylerin hızlarının etkisini yüksek tutarken, sonlara doğru bu etki giderek azalmaktadır. Böylece ikinci terimin etkisi artmaktadır. Bu sebeple birey lokal noktaya takıldığında diğer bireylerde o lokal nokta civarında toplanacaklardır. Bu durumun önüne geçebilmek için en iyi bireyin dışında popülasyona bir başka rehber birey eklenmektedir. Bu bireyin öğrenme faktörü 0-1 aralığında değişmektedir. Bu değer başlangıçta belirlenen değerden 1'e doğru artırılarak popülasyona yeni eklenen bireyin etkisi optimizasyon sürecinde giderek azalacak, en iyi bireyin etkisi ise giderek artacaktır (Yılmaz ve Küçüksille, 2015).

YA'da rastlanan diğer bir sıkıntı, 0-1 aralığında rastgele üretilen bir değerin *i.* bireyin ses şiddeti değerinden büyük olması durumunda *i.* birey lokal arama yapabilmektedir. Aksi takdirde *i.* birey uzayı keşfetmeye, uzayda yeni bölgelere yönelmeye devam etmektedir. *i.* bireye ait sinyal yayılım oranı değeri optimizasyon süresince sürekli artmakta ve optimizasyon süresince algoritmanın lokal arama kabiliyeti azalmaktadır. Bu durumun önüne geçmek için YA, yabani ot algoritması ile birleştirilmiştir. Böylece bireyler önceden belirlenen maksimum tohum sayısı kadar bulundukları çevrede çoğalmakta ve uygunluk değerleri hesaplanıp, uygunluk kalitesine göre sıralanmakta ve umut vadeden bireyler bir sonraki iterasyonda algoritmanın popülasyonunu oluşturmaktadırlar. YA'sına eklenen bu üç özellik neticesinde geliştirilen GYA, yapılan çalışmalarda daha iyi çözümler ortaya koymaktadır (Yılmaz ve Küçüksille, 2015).

3.3.6. Yapay arı kolonisi algoritması

YAK, Karaboğa tarafından arıların yiyecek arama davranışını modelleyerek geliştirilmiş bir algoritmadır (Karaboğa, 2005). Karaboğa tarafından algoritmada bazı kabuller yapılmıştır. Bunlar; her bir kaynağın nektarı sadece bir görevli arı tarafından alınmaktadır. Böylece görevli arı sayısı toplam kaynak sayısına eşittir. Görevli arıların sayısı da gözcü arıların sayısına eşittir. Nektar bittiğinde kaynağın görevli arısı artık kaşif arı olarak değerlendirilmektedir.

Kaynakların yerleri probleme ait olası çözümleri gösterirken, kaynakların nektar miktarı o kaynakla ilgili çözümlerin kalitesini göstermektedir. Algoritma en kaliteli ve fazla nektara sahip kaynağı belirlemeye çalışarak optimum çözümü bulmaya çalışmaktadır (Akay, 2009).

YAK'da, öncelikle kaşif arılar çözüm uzayına rastgele dağılarak yiyecek ararlar. Yiyecek kaynaklarını bulan kaşif arılar artık görevli arı olurlar ve kovana yiyecek taşırlar. Kovana ulaşan görevli arı nektarı boşalttıktan sonra ya kaynağa geri döner yada kaynakla ilgili konum, nektar kalitesi, nektar çokluğu, nektarın kolay çıkarılması gibi bilgileri dans aracılığıyla kovanda bekleyen gözcü arılara aktarır.

Arıların besin kaynağı hakkında bilgiyi aktarmak için dairesel dans, kuyruk dansı ve titreme dansı yapmaktadırlar. Daire dansı kaynağın uzaklığını 50-100m yakında olarak belirtirken yön ve açı bilgileri içermez. Kuyruk dansı ise 100m'den 10km'ye kadar olan alanı ifade etmektedir. Dairesel dans ise besin kaynağına ait yer ve yön bilgisini içerir. Yön bilgisi dansın açısından elde edilmektedir. Titreme dansı ise kovandaki yiyecek kapasitesini ve yiyecek getirme aktivitesini dengeleyen, arının kovandaki balı işleme görevine geçmek istediğini belirten bir danstır. Kovandaki gözcü arılar görevli arıların danslarından yiyeceğin kalitesi ile orantılı olan bir kaynağı tercih ederler.

3.3.7. Genetik algoritma

Doğal seçim ilkelerine dayanan bir arama ve optimizasyon yöntemi olarak GA'nın temel ilkeleri ilk olarak 1975 yılında Michigan Üniversitesinde John Holland tarafından ortaya konulmuştur (Zaki vd., 2015).

GA Darvin'in doğal seçim ve evrim ilkelerini temel alan sezgisel bir algoritmadır. Algoritmanın temelinde Genetik biliminin temeli olan seçim, çaprazlama ve mutasyon işlemleri vardır.

GA'da bireyler, olası her bir çözümü gösteren eşit boyutlu vektörler olarak ifade edilir. Bu vektörler genlerden oluşan kromozam olarak adlandırılır. Genler *n* boyutlu vektörlerin bir boyutunu temsil etmektedir. Kromozomu oluşturan genler için çeşitli kodlama teknikleri mevcuttur. Bunlar; ikili kodlama, permütasyon kodlama, tam sayılı kodlama ve ondalık sayılarla kodlamadır. Kullanım kolaylığı sebebiyle en yaygın kullanılanı ikili kodlama tekniğidir (Nabiyev, 2012).

Başlangıçta bireylerden rastgele seçilimle algoritmanın başında belirlenen büyüklükte ilk popülasyon oluşturulur. Oluşturulan bu popülasyondaki bireylerin ne kadar iyi olduklarını belirlemek amacıyla algoritmanın başında verilen uygunluk fonksiyonuna göre değerleri hesaplanır. Bir bireyin uygunluk değeri ne kadar iyiyse bir sonraki nesile geçebilme ihtimali o kadar yüksektir (Akpınar, 2009).

Uygunluk fonksiyonunun hesaplama neticesinde seçim işlemleriyle çaprazlama işlemi için ebeveyn bireyler tespit edilir. Seçim işlemi için rulet çemberi, rassal üreme, turnuva seçimi, sıralı seçim, Boltzman yöntemi, elitisizm yöntemi gibi birçok yöntem bulunmaktadır. En yaygın kullanılanı rulet çemberidir (Akpınar, 2009; Nabiyev, 2012).

Ebeveynler kendi aralarında belirlenen çaprazlama işlemleri ile yeni bireyi oluştururlar ve çeşitlilik sağlanmış olur. Çaprazlama işlemi için tek noktalı çaprazlama, iki noktalı çaprazlama, kısmi planlı çaprazlama, permütasyon çaprazlama, sıralı çaprazlama gibi yöntemler mevcuttur. Algoritmanın başında popülasyondaki bireylerden herhangi birinin çaprazlanma olasılığı belirtilmelidir. Eğer bu oran gereğinden küçük seçilirse çözümü geciktirebilmekte, büyük seçilirse çözümü gözden kaçırabilmektedir (Akpınar, 2009).

Oluşan yeni birey belli koşullar çerçevesinde en iyiyi bulabilmek ya da tekrardan kurtarmak amaçlı mutasyon işlemine tabi tutulabilir. İkili kodlamada mutasyon için seçilen birey terslenir. Bireylerin mutasyona uğrama olasılıkları

algoritmanın başında belirtilmelidir. Yüksek bir olasılık eldeki çözümlerin bozulmasına sebep olabilirken, düşük bir olasılık yerel çözümlere takılmaya sebep olabilir (Akpınar, 2009; Nabiyev, 2012).

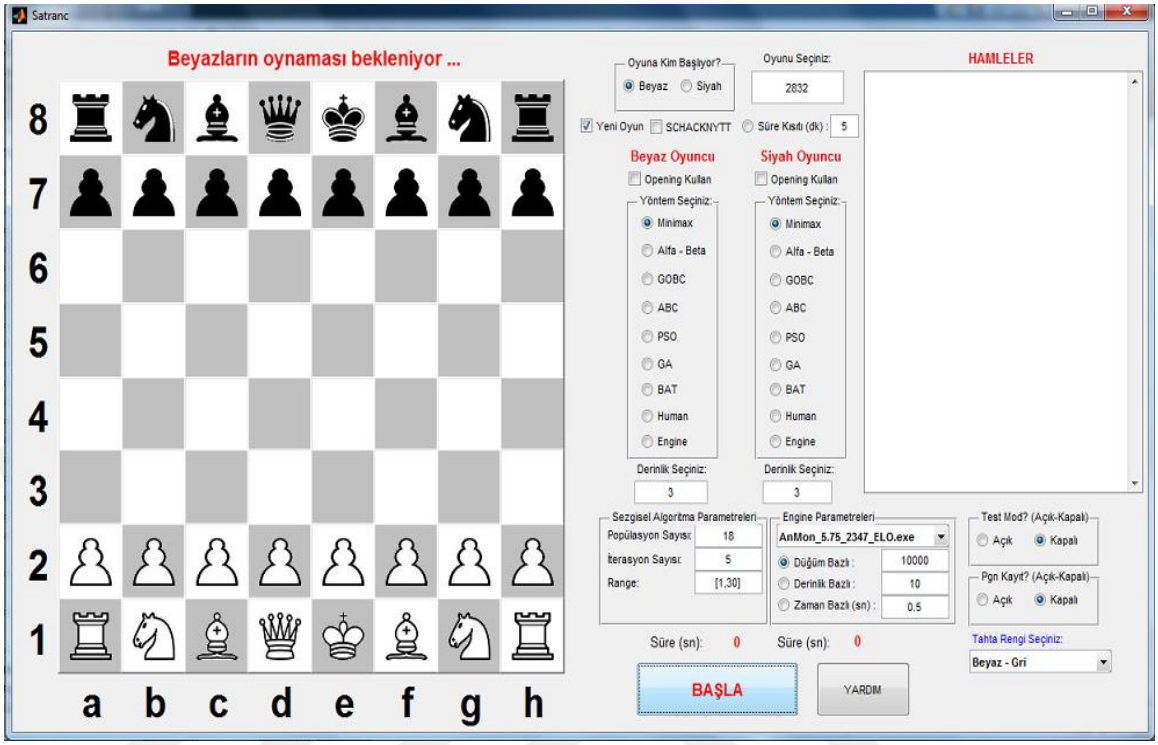
Popülasyon sayısı değişmemesi amacıyla popülasyondan eski bireyler çıkartılıp, yerine yeni elde edilen bireyler eklenir. Bunun için elitist değişim, akıllı nesil değişim, kararlı-durum değişim yöntemleri kullanılabilir (Sastry vd., 2014). Tüm bireyler uygunluk fonksiyonuna göre değerleri hesaplanıp yeni popülasyonun başarısı elde edilir. Bu işlemler algoritmanın başında belirlenen durma kriteri sağlanana kadar devam eder. Bu kriterler, istenen iterasyon sayısı, istenen başarı seviyesi ya da en iyi çözümün sabit bir değere ulaşması olabilir. En son popülasyondaki en iyi sonuç elde edilen çözümdür. Bu çözüm her zaman arama uzayındaki en iyi çözüm olmasa bile bu çözüme en yakın değer olarak kabul edilmektedir (Nabiyev, 2012).

3.4. Geliştirilen Program ve Arayüz

Bu çalışmada sezgisel yöntemlerin, satranç motorlarının ve insanın birbirleriyle satranç oyunu oynayabileceği ve performanslarını karşılaştırabilecekleri bir yazılım gerçekleştirilmiştir. Yazılım görsel olarak taşların hareketlerini ekranda sunarken aynı zamanda insan bir kullanıcının oynaması durumunda taşların seçilip hareket ettirilebilmesini gerçekleştirmektedir. Yazılım, oyun sırasının kimde olduğu ve oyunun sona erme bilgilerinin yanında, pgn standardında ve klasik standartta oyun notasyon bilgilerini kullanıcıya sunabilmektedir.

Açılış hamlelerini içeren veritabanlarının seçime göre kullanılabilmesi ve farklı test setlerinde performans değerlendirmelerinin yapılması da programın artı yönlerindendir. Herhangi klasik satranç iletişim protokolü (UCI) kullanan bir satranç oyun motorunun programa ait alt klasöre yerleştirilmesiyle geliştirilen yazılım bu oyun motoruyla iletişime geçip karşılıklı maç yapabilmektedir. Sezgisel algoritmaların ve satranç motorlarının parametrelerinin ayarlanması, tahta renginin değiştirilebilmesi de yazılımın diğer özellikleri arasında

sayılabilir. Matlab R2014a programında yazılan koda ait arayüz Şekil 3.12'de verilmiştir.



Şekil 3.12. Geliştirilen programa ait arayüz

Şekil 3.12'de görüldüğü gibi programda sezgisel yöntemler kendi aralarında, satranç motorlarına karşı yada insana karşı oyun oynayabilmektedir. UCI veri tabanından alınan oyunlarda oyuna başlayacak oyuncu en üstte bulunan "Beyaz" ve "Siyah" radyo butonları ile seçilebilmektedir. Ayrıca kullanıcı sıfırdan bir oyun başlatabildiği gibi "SCHACKNYTT" veritabanında bulunan ünlü satranç oyuncularının oyunlarına ait oyunu başlatabilmektedir. Oyuna ait hamleler arayüzün sağ tarafında "Hamleler" kısmında gösterilmektedir. Siyah ve beyaz oyuncu için istenilen yöntem "Yöntem seçiniz" kısmından seçilebilmektedir. Ayrıca sıfır oyunlar için veritabanından ulaşabildiği sürece ilk 20 hamlenin veritabanından seçilebileceği "açılış (opening) kullan" seçeneği işaretlenebilmektedir. Sezgisel algoritma parametrelerinin seçilebildiği bir sekme ve satranç motoru parametrelerinin seçilebildiği sekmelerde mevcuttur. Ayrıca, oynanan oyunların sonuçlarının ve sonuçların uluslararası satranç notasyonlarının kaydedilebileceği "Test Mod" ve "Pgn Kayıt" kısımları

mevcuttur. Satranç tahtasının rengi beyaz-gri ve açık kahve-koyu kahve olarak ayarlanabilmektedir.

4. ARAŞTIRMA BULGULARI VE TARTIŞMA

Bu çalışmada geliştirilen yöntem minimaks ve α-β gibi tüm hamleleri tarayabilen yöntemler ile kıyaslanmıştır. Bunun yanı sıra yöntemin arama mekanizmasında test edilen KTGO, YAK, PSO, GA ve GYA yöntemleri de kendi aralarında test edilebilmiştir. Bu sayılan yöntemler UCI veritabanından elde edilen bitime belli hamle kala mat durumlarını içeren veritabanı ve önceden oynanmış popüler oyunlara ait hamleleri içeren SCHACKNYTT veritabanında bulunan 16 oyun durumlarında test edilmiştir.

4.1. Chess Veritabanı Üzerinde Yapılan Çalışmalar

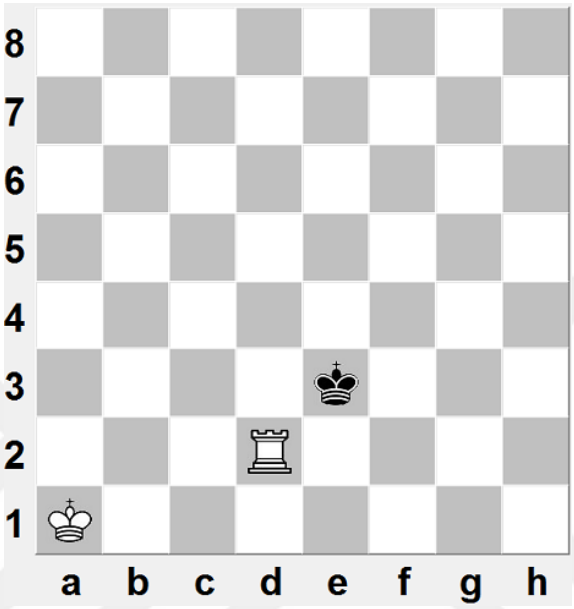
Chess veritabanı popüler UCI örnek veritabanında yer alan bitime sayılı hamle kalan mat durumlarını içerir. Bu veritabanında ayrı ayrı problemler olarak 1 hamlede pat, 2 hamlede, 3 hamlede mat örneklerinden dörder adet ve 5 hamlede mat örneğinden bir adet kullanılmıştır. Bu örneklerde özellikle 5 hamlede mat durumları için derinlik 5 ve üzerinde derinlik içermesi sebebiyle kısmen zor problemler kategorisindedir. Bunun sebepleri arasında taş sayısının az olmasına rağmen kazanılacak hamlenin oyunun ileriki hamlelerinde, yani daha yüksek derinliklerde olmasıdır. Minimaks ve α-β yöntemleri bu problemlerde aşırı yavaş olmakla birlikte hiç hesaplayamaya da bilmektedirler.

27. oyun, 1327. oyun, 2015. oyun, 2756. oyun, 2825. oyun, 2845. oyun, 2873. oyun, 2900. oyun, 3000. oyun, 3047. oyun, 3098. oyun, 3140. oyun ve 3300. oyun için beşer defa çalıştırılan minimaks, α-β, KTGO, YAK, PSO, GA ve GYA ile elde edilen sonuçlar analiz edildiğinde yöntemlerin başarısını ölçmek için kullanılan kriterlerden olan minimum değer, maksimum değer, ortalama ve standart sapma gibi ölçütler karşılaştırılmıştır.

Chess veritabanında yer alan oyunlar için sezgisel algoritmaların parametrelerinden; popülasyon değeri 60, iterasyon değeri 20 ve sınır değeri [1-20] olarak belirlenmiş olup, oyuna 1 hamlede mat için rakip kabul edilen

beyaz oyuncu, diğer tüm oyunlar için rakip kabul edilen siyah oyuncu başlar ve yöntem olarak α-β kullanmaktadır.

1 hamlede pat işlemi için rastgele seçilen 27. oyuna ait durum Şekil 4.1'de gösterilmektedir.



Şekil 4.1. Chess veritabanında yer alan 27. oyun

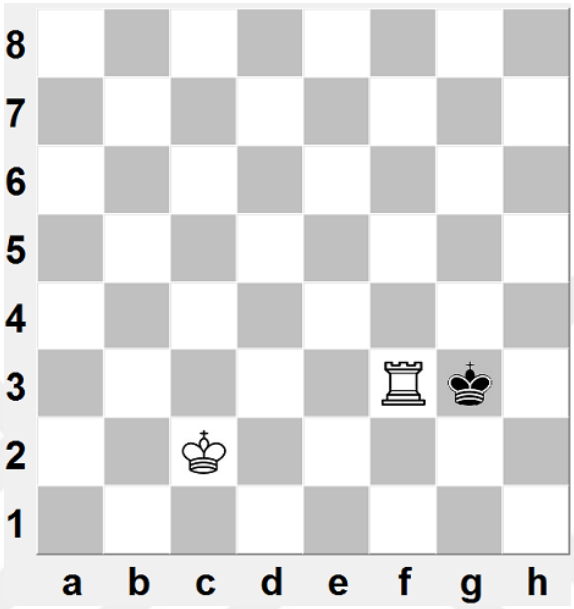
Şekil 4.1'de verilen oyuna ait yöntemlerin başarısını ölçmek için kullanılan ölçütlerden olan maksimum değer (maks), minimum değer (min), ortalama (ort) ve standart sapma (SSp) Çizelge 4.1'de verilmiştir.

Çizelge 4.1. 27. Oyuna ait puan ve süre değerleri

	Puan				Süre (Sn)			
	Maks	**Min**	**Ort**	**SSp**	**Maks**	**Min**	**Ort**	**SSp**
Minimaks	0	0	0	0	12,4793	0,7005	5,6100	4.4354
α-β	0	0	0	0	19,6751	4,6547	9,5699	5,8763
KTGO	1	1	1	0	13,2546	12,2778	12,6923	0,3912
YAK	1	1	1	0	7,7429	6,6894	7,1659	0,4557
PSO	1	1	1	0	11,0341	9,6197	10,5372	0,5972
GA	1	1	1	0	7,2029	4,5283	5,5206	1,0086
GYA	1	1	1	0	7,4432	6,6781	7,1094	0,2921

Çizelge 4.1'de oyun 1 hamlede pat olup rakibin ve yöntemin derinlik değeri 2'dir. Rakibin yöntemi ise belirtildiği gibi α-β'dır.

1 hamlede pat işlemi için rastgele seçilen 1327. oyuna ait durum Şekil 4.2'de gösterilmektedir.



Şekil 4.2. Chess veritabanında yer alan 1327. oyun

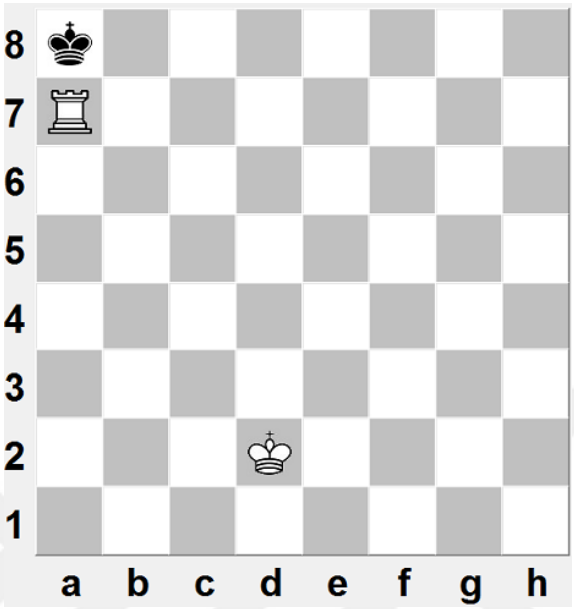
Şekil 4.2'de verilen oyuna ait yöntemlerin başarı değerleri Çizelge 4.2'de verilmiştir.

Çizelge 4.2. 1327. Oyuna ait puan ve süre değerleri

	Puan				Süre (Sn)			
	Maks	Min	Ort	SSp	Maks	Min	Ort	SSp
Minimaks	0	0	0	0	6,1282	3,1560	4,7333	1,2946
α-β	0	0	0	0	22,1994	4,4081	13,7348	7,1067
KTGO	1	1	1	0	10,3735	8,5681	9,4569	0,6656
YAK	1	1	1	0	7,1803	5,3165	6,0837	0,7291
PSO	1	1	1	0	14,5328	11,8439	13,2006	1,1549
GA	1	1	1	0	9,6609	6,1075	8,6147	1,4657
GYA	1	1	1	0	9,9191	8,3801	9,2787	0,6802

Çizelge 4.2'de oyun 1 hamlede pat olup rakibin ve yöntemin derinlik değeri 2'dir.

1 hamlede pat işlemi için rastgele seçilen 2015. oyuna ait durum Şekil 4.3'de gösterilmektedir.



Şekil 4.3. Chess veritabanında yer alan 2015. oyun

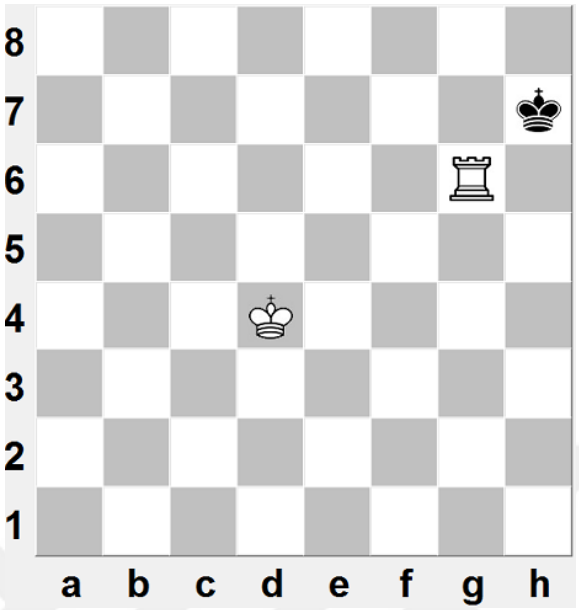
Şekil 4.3'de verilen oyuna ait puan ve süre değerleri gibi ölçütler Çizelge 4.3'de verilmiştir.

Çizelge 4.3. 2015. Oyuna ait puan ve süre değerleri

	Puan				Süre (Sn)			
	Maks	**Min**	**Ort**	**SSp**	**Maks**	**Min**	**Ort**	**SSp**
Minimaks	0	0	0	0	0,1823	0,1530	0,1673	0,0106
α-β	0	0	0	0	0,6890	0,3515	0,4252	0,1476
KTGO	1	1	1	0	7,6363	7,0404	7,3774	0,2457
YAK	1	1	1	0	5,1687	3,0097	3,5900	0,8996
PSO	1	1	1	0	7,9932	7,0618	7,7458	0,3889
GA	1	1	1	0	3,1212	2,6085	2,8856	0,1962
GYA	1	1	1	0	7,4789	6,1749	6,8104	0,4832

Çizelge 4.3'de oyun 1 hamlede pat olup rakibin ve yöntemin derinlik değeri 2'dir.

1 hamlede pat işlemi için rastgele seçilen 2756. oyuna ait durum Şekil 4.4'de gösterilmektedir.



Şekil 4.4. Chess veritabanında yer alan 2756. oyun

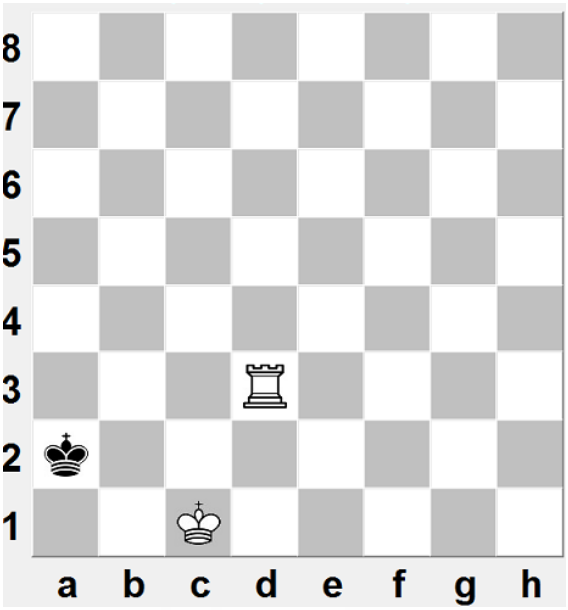
Şekil 4.4'de verilen oyuna ait yöntemlerin başarısını ölçmek için kullanılan kriterler Çizelge 4.4'de verilmiştir.

Çizelge 4.4. 2756. Oyuna ait puan ve süre değerleri

	Puan				Süre (Sn)			
	Maks	**Min**	**Ort**	**SSp**	**Maks**	**Min**	**Ort**	**SSp**
Minimaks	0	0	0	0	0,1047	0,0983	0,1016	0,0028
α-β	0	0	0	0	0,1657	0,1541	0,1575	0,0048
KTGO	1	1	1	0	7,9656	7,2562	7,6231	0,3202
YAK	1	1	1	0	4,5655	2,6884	3,2011	0,7765
PSO	1	1	1	0	8,8003	7,2100	8,1130	0,6060
GA	1	1	1	0	3,1725	2,7163	2,9766	0,1999
GYA	1	1	1	0	7,2942	6,0211	6,8032	0,4801

Çizelge 4.4'te oyun 1 hamlede pat olup rakibin ve yöntemin derinlik değeri 2'dir.

2 hamlede mat işlemi için rastgele seçilen 2825. oyuna ait durum Şekil 4.5'de gösterilmektedir.



Şekil 4.5. Chess veritabanında yer alan 2825. oyun

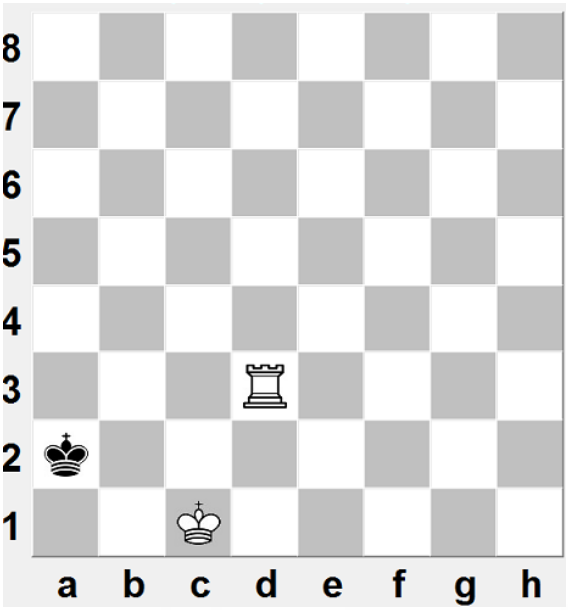
Şekil 4.5'de verilen oyuna ait puan ve süre değerleri Çizelge 4.5'de verilmiştir.

Çizelge 4.5. 2825. Oyuna ait puan ve süre değerleri

	Puan				Süre (Sn)			
	Maks	**Min**	**Ort**	**SSp**	**Maks**	**Min**	**Ort**	**SSp**
Minimaks	1	1	1	0	0,4333	0,4215	0,4273	0,0046
α-β	1	1	1	0	1,4524	1,4246	1,4432	0,0115
KTGO	1	1	1	0	29,4283	27,7824	28,7377	0,6665
YAK	1	1	1	0	21,5695	19,8306	20,7317	0,6727
PSO	1	1	1	0	22,4091	21,0137	21,7561	0,5378
GA	1	1	1	0	22,1126	20,9442	21,4157	0,4393
GYA	1	1	1	0	20,4534	19,5688	20,0383	0,3413

Çizelge 4.5'te oyun 2 hamlede mat olup rakibin ve yöntemin derinlik değeri 3'tür.

2 hamlede mat işlemi için rastgele seçilen 2845. oyuna ait durum Şekil 4.6'da gösterilmektedir.



Şekil 4.6. Chess veritabanında yer alan 2845. oyun

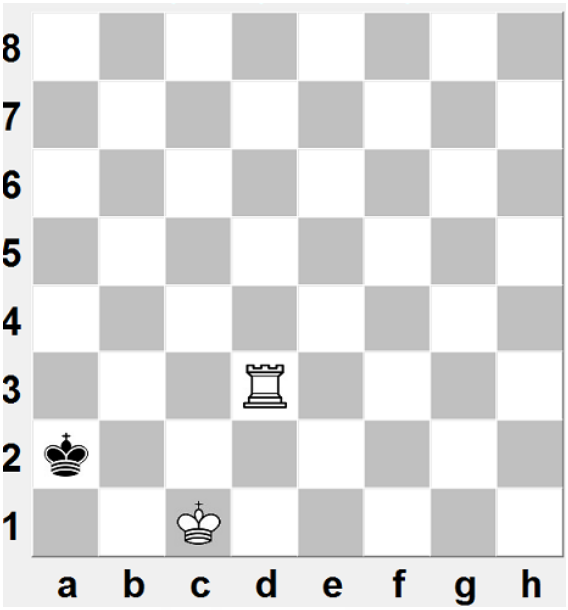
Şekil 4.6'da verilen oyuna ait yöntemlerin başarısını ölçmek için kullanılan kriterler Çizelge 4.6'da verilmiştir.

Çizelge 4.6. 2845. Oyuna ait puan ve süre değerleri

	Puan				Süre (Sn)			
	Maks	**Min**	**Ort**	**Std**	**Maks**	**Min**	**Ort**	**SSp**
Minimaks	1	1	1	0	1,1012	1,0103	1,0458	0,0344
α-β	1	1	1	0	2,3977	2,3398	2,3715	0,0252
KTGO	1	1	1	0	42,3185	40,1446	41,3404	0,8245
YAK	1	1	1	0	28,6849	24,2197	26,2966	1,7875
PSO	1	1	1	0	28,7914	28,1148	28,4988	0,2804
GA	1	1	1	0	27,9018	27,0376	27,5318	0,3160
GYA	1	1	1	0	27,1544	24,5094	25,9946	1,0018

Çizelge 4.6'da oyun 2 hamlede mat olup rakibin ve yöntemin derinlik değeri 3'tür.

2 hamlede mat işlemi için rastgele seçilen 2873. oyuna ait durum Şekil 4.7'de gösterilmektedir.



Şekil 4.7. Chess veritabanında yer alan 2873. oyun

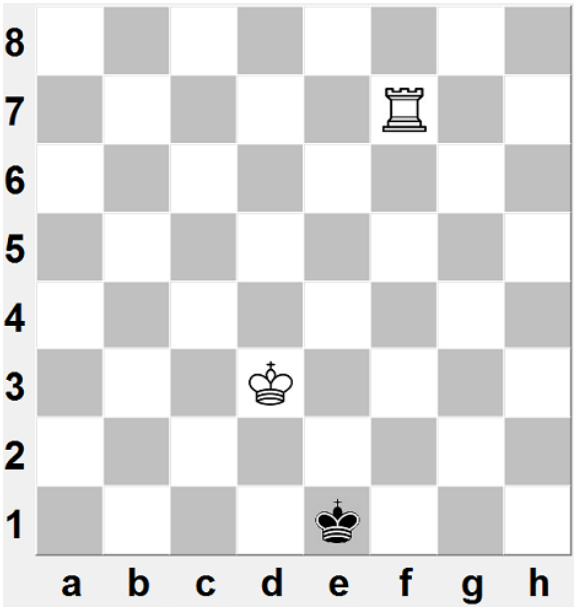
Şekil 4.7'de verilen oyuna ait sonuçlar Çizelge 4.7'de verilmiştir.

Çizelge 4.7. 2873. Oyuna ait puan ve süre değerleri

	Puan				Süre (Sn)			
	Maks	**Min**	**Ort**	**SSp**	**Maks**	**Min**	**Ort**	**SSp**
Minimaks	1	1	1	0	1,0267	0,9404	0,9651	0,0362
α-β	1	1	1	0	2,1588	2,1456	2,1520	0,0047
KTGO	1	1	1	0	37,5202	36,4145	36,8749	0,4289
YAK	1	1	1	0	26,9750	24,9755	25,8877	0,7171
PSO	1	1	1	0	28,5673	27,6538	28,1669	0,3301
GA	1	1	1	0	28,1155	27,0536	27,6775	0,3850
GYA	1	1	1	0	25,6499	24,0592	24,9641	0,6825

Çizelge 4.7'de oyun 2 hamlede mat olup rakibin ve yöntemin derinlik değeri 3'tür.

2 hamlede mat işlemi için rastgele seçilen 2900. oyuna ait durum Şekil 4.8'de gösterilmektedir.



Şekil 4.8. Chess veritabanında yer alan 2900. oyun

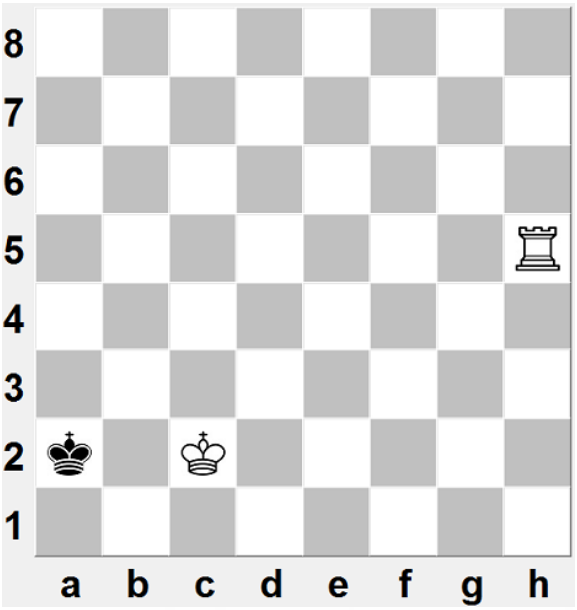
Şekil 4.8'de verilen oyuna ait yöntemlerin başarısını ölçmek için kullanılan kriterlerden olan minimum değer, maksimum değer, ortalama ve standart sapma gibi ölçütler Çizelge 4.8'de verilmiştir.

Çizelge 4.8. 2900. Oyuna ait puan ve süre değerleri

	Puan				Süre (Sn)			
	Maks	**Min**	**Ort**	**SSp**	**Maks**	**Min**	**Ort**	**SSp**
Minimaks	1	1	1	0	1,0218	1,0026	1,0136	0,0082
α-β	1	1	1	0	2,1166	2,0417	2,0621	0,0364
KTGO	1	1	1	0	3,4376	3,3190	3,3788	0,0445
YAK	1	1	1	0	27,4712	23,9828	25,5469	1,7511
PSO	1	1	1	0	29,8178	28,7893	29,2838	0,4888
GA	1	1	1	0	29,2377	28,2024	28,6199	0,4585
GYA	1	1	1	0	23,2811	22,7260	23,0104	0,2266

Çizelge 4.8'de oyun 2 hamlede mat olup rakibin ve yöntemin derinlik değeri 3'tür.

3 hamlede mat işlemi için rastgele seçilen 3000. oyuna ait durum Şekil 4.9'da gösterilmektedir.



Şekil 4.9. Chess veritabanında yer alan 3000. oyun

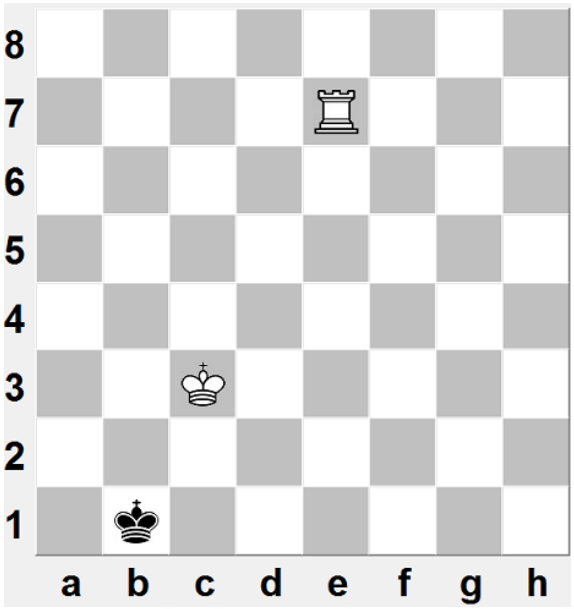
Şekil 4.9'da verilen oyuna ait sonuçlar Çizelge 4.9'da verilmiştir.

Çizelge 4.9. 3000. Oyuna ait puan ve süre değerleri

	Puan				Süre (Sn)			
	Maks	**Min**	**Ort**	**SSp**	**Maks**	**Min**	**Ort**	**SSp**
Minimaks	1	1	1	0	25,4008	25,3657	25,3854	0,0176
α-β	1	1	1	0	16,3288	16,2478	16,2889	0,0320
KTGO	1	0	0,6	0,5477	328,7194	121,8682	197,2661	77,5522
YAK	0	0	0	0	1,729e+3	108,9968	729,0707	738,007
PSO	1	1	1	0	79,4984	37,5614	61,8985	21,5070
GA	0	0	0	0	296,0641	130,0870	211,5758	83,0291
GYA	1	1	1	0	86,1858	79,4789	83,7718	2,9687

Çizelge 4.9'da oyun 3 hamlede mat olup rakibin derinlik değeri 3 ve yöntemin derinlik değeri 4'tür.

3 hamlede mat işlemi için rastgele seçilen 3047. oyuna ait durum Şekil 4.10'da gösterilmektedir.



Şekil 4.10. Chess veritabanında yer alan 3047. oyun

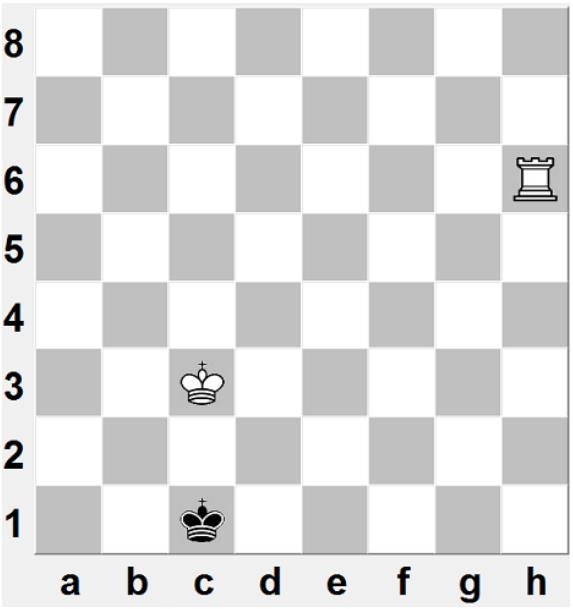
Şekil 4.10'da verilen oyuna ait yöntemlerin birbirleri ile kıyaslanmaları için minimum değer, maksimum değer, ortalama ve standart sapma gibi ölçütler Çizelge 4.10'da verilmiştir.

Çizelge 4.10. 3047. Oyuna ait puan ve süre değerleri

	Puan				Süre (Sn)			
	Maks	**Min**	**Ort**	**SSp**	**Maks**	**Min**	**Ort**	**SSp**
Minimaks	1	1	1	0	11,3526	10,9541	11,0506	0,1695
α-β	1	1	1	0	6,4299	6,1978	6,2741	0,1080
KTGO	1	1	1	0	5,4026	4,5907	4,9692	0,3290
YAK	1	0	0,4	0,5477	230,1300	31,8183	128,7173	95,0390
PSO	1	0	0,8	0,4472	226,0681	42,4768	79,7617	81,7899
GA	1	1	1	0	44,1550	42,4117	43,3844	0,6312
GYA	1	0	0,8	0,4472	172,6716	30,4959	61,7789	62,0595

Çizelge 4.10'da oyun 3 hamlede mat olup rakibin derinlik değeri 3 ve yöntemin derinlik değeri 4'tür.

3 hamlede mat işlemi için rastgele seçilen 3098. oyuna ait durum Şekil 4.11'de gösterilmektedir.



Şekil 4.11. Chess veritabanında yer alan 3098. oyun

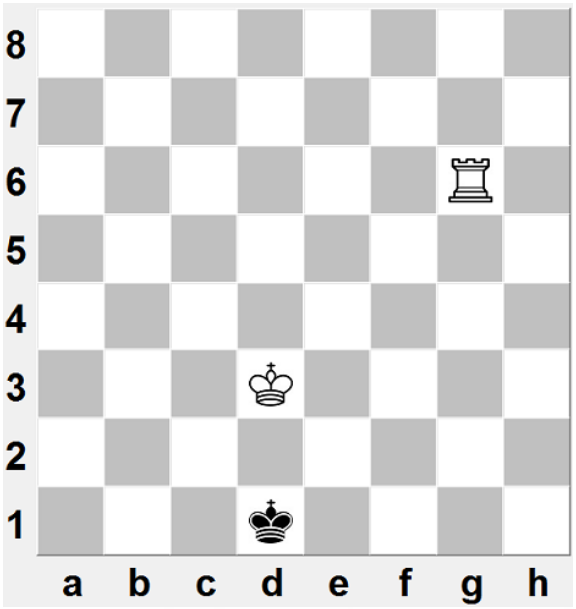
Şekil 4.11'deki oyuna ait değerler Çizelge 4.11'de verilmiştir.

Çizelge 4.11. 3098. Oyuna ait puan ve süre değerleri

	Puan				Süre (Sn)			
	Maks	**Min**	**Ort**	**SSp**	**Maks**	**Min**	**Ort**	**SSp**
Minimaks	1	1	1	0	27,1532	24,3259	26,4702	1,2044
α-β	1	0	0,6	0,5477	348,4219	12,7752	118,6787	153,103
KTGO	1	0	0,6	0,5477	236,3101	65,2480	115,6506	73,5267
YAK	1	0	0,4	0,5477	462,8763	78,4597	258,2578	191,428
PSO	0	0	0	0	1,245e+4	174,798	3,452e+3	5,219e+3
GA	0	0	0	0	628,7156	193,1632	399,9653	195,1468
GYA	1	0	0,6	0,5477	221,6429	74,7248	115,0920	61,5998

Çizelge 4.11'de oyun 3 hamlede mat olup rakibin derinlik değeri 3 ve yöntemin derinlik değeri 4'tür.

3 hamlede mat işlemi için rastgele seçilen 3140. oyuna ait durum Şekil 4.12'de gösterilmektedir.



Şekil 4.12. Chess veritabanında yer alan 3140. oyun

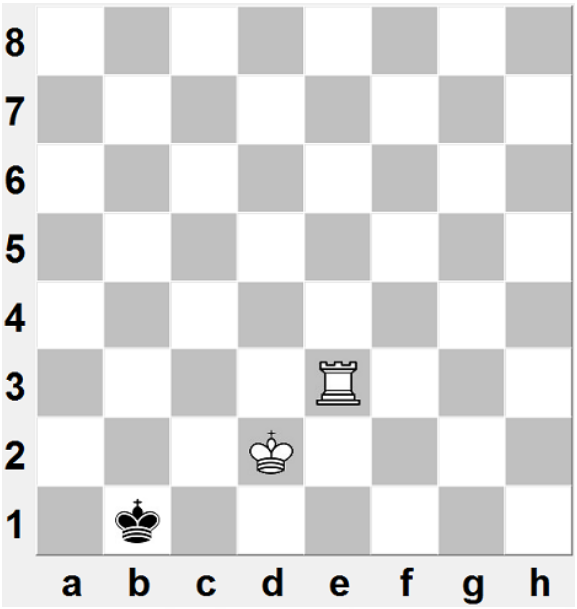
Şekil 4.12'de verilen oyuna ait sonuçlar Çizelge 4.12'de verilmiştir.

Çizelge 4.12. 3140. Oyuna ait puan ve süre değerleri

	Puan				Süre (Sn)			
	Maks	**Min**	**Ort**	**SSp**	**Maks**	**Min**	**Ort**	**SSp**
Minimaks	1	1	1	0	31,5820	27,0101	29,2056	2,1000
α-β	1	0	0,4	0,5477	288,4278	13,0806	95,6641	116,6491
KTGO	1	1	1	0	95,1577	72,6306	84,7741	11,3220
YAK	1	0	0,6	0,5477	139,9849	96,7907	115,1742	22,3052
PSO	0	0	0	0	543,9723	203,2735	410,5133	136,4752
GA	1	0	0,2	0,4472	399,8765	110,9725	241,6698	106,2471
GYA	1	1	1	0	86,3624	84,6027	85,6278	0,8620

Çizelge 4.12'de oyun 3 hamlede mat olup rakibin derinlik değeri 3 ve yöntemin derinlik değeri 4'tür. Oyunlardan 3140 için elde edilen sonuçlar incelendiğinde KTGO ve GYA'nın diğer yöntemleri geride bırakıp her seferinde belirlenen oyunu başarı ile tamamladığı görülmektedir. Süre açısından bakıldığında ise diğer yöntemlerden daha kısa sürede sonuca ulaşmışlardır.

5 hamlede mat işlemi için rastgele seçilen 3300. oyuna ait durum Şekil 4.13'de gösterilmektedir.



Şekil 4.13. Chess veritabanında yer alan 3300. oyun

Şekil 4.13'de verilen oyuna ait yöntemlerin başarısını ölçmek için kullanılan kriterlerden olan minimum değer, maksimum değer, ortalama ve standart sapma gibi ölçütler Çizelge 4.13'de verilmiştir.

Çizelge 4.13. 3300. Oyuna ait puan ve süre değerleri

	Puan				Süre (Sn)			
	Maks	**Min**	**Ort**	**SSp**	**Maks**	**Min**	**Ort**	**SSp**
Minimaks	Hesaplayamadı							
α-β	Hesaplayamadı							
KTGO	1	1	1	0	78,5166	46,1918	61,1169	13,2033
YAK	1	0	0,6	0,5477	2,278e+3	794,250	1,413e+3	582,8286
PSO	0	0	0	0	2,092e+3	1,449e+3	1,835e+3	352,3531
GA	1	0	0,2	0,4472	9,097e+3	1,315e+3	3,602e+3	3,151e+3
GYA	1	0	0,6	0,5477	2,345e+3	843,793	1,869e+3	428,7635

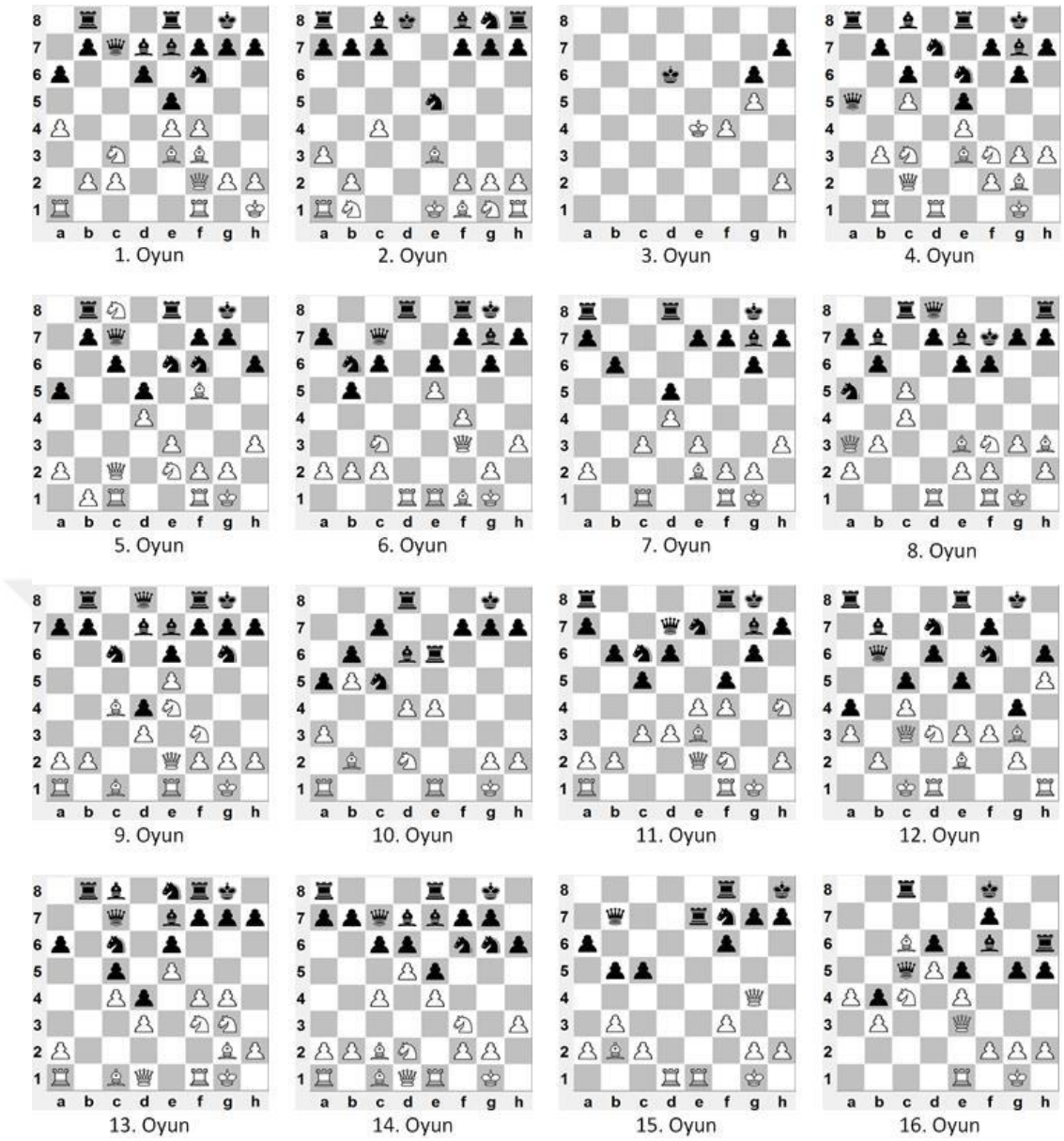
Çizelge 4.13'de oyun 5 hamlede mat olup rakibin derinlik değeri 3 ve yöntemin derinlik değeri 7'dir. Bu oyun için sezgisel algoritmaların popülasyon değeri 100, iterasyon değeri 50 ve sınır değeri [1-50] belirlenmiştir. Oyuna siyah oyuncu başlayıp yöntem olarak α-β kullanmaktadır. Derinliğin yüksek olduğu bu oyunda KTGO'nun tüm oyunları kazandığı ve diğer yöntemleri hem puan olarak hem de süre olarak geçtiği görülmektedir.

Çizelgeler incelendiğinde ilk başta minimaks ve α-β belli bir derinliğin üzerinde hesaplama yapamadığı göze çarpmakta olup bu problemlerin çözümünde sezgisel algoritmalara ihtiyaç duyulmuştur.

4.2. SCHACKNYTT Veritabanı Üzerinde Yapılan Çalışmalar

Swedish chessmagazine SCHACKNYTT (chessnews)'de yayınlanan ünlü satranç oyuncularının oynadığı popüler oyunlardaki stratejik hamlelerin tahminini baz alıp bu problemler için düşünülmüş değer hesaplama yöntemlerine göre algoritmaların puanları hesaplanmıştır. Bu sayede yöntemlerin birbirileriyle kıyaslanmasının yanı sıra CRAFTY gibi yüksek ELO değerine sahip ünlü satranç programlarına ne derece yakın olduğu tespit edilebilmiştir. Bu veritabanına ait örneklerden 1. oyun, ünlü satranç ustalarından olan Kasparov'un 1985 yılında Karpov ile oynadığı oyunun 17. hamlede oynanması gereken en iyi hamleyi tespit etmeyi içerir. Her bir örnekte olduğu gibi bu problemde de belirlenen hamlede yapılacak tercihe göre farklı puanlar ayarlanmıştır. Tüm oyunlara ait durumlar Şekil 4.14'te gözükmektedir.

SCHACKNYTT veritabanındaki 16 oyun için beşer defa çalıştırılan minimaks, α-β, KTGO, YAK, PSO, GA ve GYA sonucunda 560 adet sonuç analiz edildiğinde yöntemlerin başarısını ölçmek için kullanılan kriterlerden olan toplam 16 oyununun minimum, maksimum, ortalama değerlerinin toplamı ve standart sapması Çizelge 4.14'de verilmiştir.



Şekil 4.14. SCHACKNYTT veritabanında yer alan 16 oyun

Çizelge 4.14. SCHACKNYTT oyunu sonuçları

	Max	**Min**	**Ortalama**	**Standart Sapma**
Minimaks	32	5	11,8	2,0669
α-β	40	26	30,6	2,9044
KTGO	91	31	68,2	4,0961
YAK	78	34	48,6	4,1930
PSO	85	16	45,6	3,9042
GA	77	29	52,4	4,1764
GYA	80	26	51,5	4,3117

Çizelge 4.14'e göre maksimumda alınabilecek puan Çizelge 4.15'deki ünlü oyun motorları ile kıyaslandığında yöntemlerin listeye girmeyi başardığı, KTGO'nun ise üst sıralarda yer aldığı görülmüştür. Popülasyon ve iterasyonun artırılması başarı oranını biraz daha artıracağı öngörülmektedir.

Çizelge 4.15. Ünlü satranç oyun motorlarının ve programdaki yöntemlerin SCHACKNYTT'den aldıkları puanlar

	Puan
Deep Sjeng 1.5a Athlon 1200	98
Chess Tiger 14.9 Palm 42MHz 16MB	98
Rebel 8 P200 MMX	98
Junior 3.5 P90	97
Shredder 5 AMD K6-2 450	96
Nimzo 99 AMD K6-2 450	96
Chessmaster 6000 AMD K6-2 450	95
Rebel Century AMD K6-2 450	94
Genius 5 P200 MMX	94
KTGO	91
MChess Pro MMX	91
Junior 8 Athlon 1200	90
Genius 5 P90	90
Shredder 5.32 Athlon 1200	89
Patzer 3.11 AMD K6-2 450	89
Hiarcs 4 P90	89
Fritz 3 P90	87
Nimzo 7.32 AMD K6-2 450	86
PSO	85
Hiarcs 6 P90	85
Rebel Century 4 Athlon 1200	83
MChess Pro 5 P90	81
YAK	78
GA	77
CRAFTY 17.04 AMD K6-2 450	76
Rebel Century 3 AMD K6-2 450	75
GYA	74
EXchess 2.51 AMD K6-2 450	74
α-β	40
Minimaks	32

Çizelge 4.16. Ünlü satranç oyun motorlarının ve programda yer alan yöntemlerin ilk 8 SCHACKNYTT durumlarından aldıkları puanlar

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Deep Sjeng 1.5a Athlon 1200	-	7	10	7	7	9	7	10
Chess Tiger 14.9 Palm 42MHz 16MB	9	7	-	7	7	9	7	-
Rebel 8 P200 MMX	-	7	10	7	7	9	7	5
Junior 3.5 P90	4	9	-	7	7	10	8	5
Shredder 5 AMD K6-2 450	-	4	10	7	7	10	10	-
Nimzo 99 AMD K6-2 450	10	7	10	7	10	10	7	5
Chessmaster 6000 AMD K6-2 450	-	9	10	7	7	9	7	10
Rebel Century AMD K6-2 450	-	7	10	7	7	10	7	5
Genius 5 P200 MMX	-	7	10	7	7	5	7	5
MChess Pro MMX	-	9	10	7	7	-	7	5
Junior 8 Athlon 1200	-	10	10	7	7	10	10	10
Genius 5 P90	-	7	10	7	7	5	7	5
Shredder 5.32 Athlon 1200	-	9	10	7	7	10	10	-
Patzer 3.11 AMD K6-2 450	-	7	10	8	7	10	10	--
Hiarcs 4 P90	-	7	-	8	7	10	7	-
Fritz 3 P90	9	9	-	7	7	10	7	5
Nimzo 7.32 AMD K6-2 450	-	-	10	7	7	10	7	5
Hiarcs 6 P90	-	7	10	7	7	10	7	-
Rebel Century 4 Athlon 1200	-	7	10	7	7	10	7	5
MChess Pro 5 P90	-	9	10	7	7	9	7	-
CRAFTY 17.04 AMD K6-2 450	-	9	-	8	7	5	7	5
Rebel Century 3 AMD K6-2 450	-	7	10	7	7	10	7	5
EXchess 2.51 AMD K6-2 450	-	9	10	7	7	-	7	-
Minimaks	-	-	-	7	7	-	-	-
α-β	-	-	-	3	7	-	-	-
KTGO	3	7	10	4	-	9	10	-
YAK	5	-	10	-	-	9	10	-
PSO	3	7	10	-	-	9	10	-
GA	3	-	10	-	-	9	10	-
GYA	6	-	8	-	-	9	10	-

Çizelge 4.17. Ünlü satranç oyun motorlarının ve programda yer alan yöntemlerin son 8 SCHACKNYTT durumlarından aldıkları puanlar

	9	**10**	**11**	**12**	**13**	**14**	**15**	**16**
Deep Sjeng 1.5a Athlon 1200	-	-	5	7	8	4	10	7
Chess Tiger 14.9 Palm 42MHz 16MB	8	-	10	7	6	4	10	7
Rebel 8 P200 MMX	10	-	10	7	8	4	-	7
Junior 3.5 P90	10	3	7	10	-	10	-	7
Shredder 5 AMD K6-2 450	10	-	-	8	8	5	10	7
Nimzo 99 AMD K6-2 450	-	-	-	7	6	5	4	7

Çizelge 4.17. Ünlü satranç oyun motorlarının ve programda yer alan yöntemlerin son 8 SCHACKNYTT durumlarından aldıkları puanlar (Devam)

Chessmaster 6000 AMD K6-2 450	8	10	-	-	1	-	10	7
Rebel Century AMD K6-2 450	10	-	-	7	8	-	9	7
Genius 5 P200 MMX	10	-	-	7	4	8	10	7
MChess Pro MMX	-	-	10	7	8	4	10	7
Junior 8 Athlon 1200	-	-	-	10	9	-	-	7
Genius 5 P90	10	-	-	7	-	8	10	7
Shredder 5.32 Athlon 1200	-	-	-	8	8	4	9	7
Patzer 3.11 AMD K6-2 450	-	10	5	7	8	-	-	7
Hiarcs 4 P90	10	10	5	-	4	4	10	7
Fritz 3 P90	6	-	5	10	1	4	-	7
Nimzo 7.32 AMD K6-2 450	-	-	7	7	4	5	10	7
Hiarcs 6 P90	10	-	5	7	4	4	-	7
Rebel Century 4 Athlon 1200	8	-	-	7	8	-	-	7
MChess Pro 5 P90	-	-	-	7	8	-	10	7
CRAFTY 17.04 AMD K6-2 450	10	-	-	7	6	5	-	7
Rebel Century 3 AMD K6-2 450	-	-	-	7	8	-	-	7
EXchess 2.51 AMD K6-2 450	9	10	-	3	-	5	-	7
Minimaks	7	-	5	-	-	5	4	5
α-β	10	-	-	-	-	5	-	7
KTGO	8	10	5	-	8	5	5	7
YAK	9	10	-	-	9	9	-	7
PSO	5	10	7	6	9	-	9	-
GA	9	10	7	5	9	5	-	-
GYA		10	6	7	9	-	9	-

Çizelge 4.16 ve Çizelge 4.17'de satranç motorları ve kullanılan yöntemlerin sırayla 16 oyundan aldıkları puanlar yer almaktadır. Çizelge 4.15'de ise, 16 oyundan alınan toplam puanlar verilmiştir. Çizelgelere göre, ünlü oyun motorları ile kıyaslandığında yöntemlerin listeye girmeyi başardığı, KTGO'nun ise üst sıralarda yer aldığı görülmüştür.

4.3. Sezgisel Yöntemlerin Birbirleriyle Yaptığı Maçların Karşılaştırılması

Bu deneysel çalışmada sezgisel yöntemlerin kıyaslanabilmesi amacıyla sıfırdan başlayan bir oyunda yöntemler üçer defa birbirleriyle oynatılmıştır. Sezgisel yöntemlerin birbiriyle yaptığı maçlarda FIDE'nin belirlediği ilk 40 hamle 90 dakika, sonraki hamleler 30 dakika kuralı göz önünde bulundurulmuştur. Çalışmada açılış hamleleri (opening book) olarak Tarrasch satranç arayüzü

kullanılmıştır (Forster, 2015). Parametreler 3 derinlikte, 36 popülasyon, 20 iterasyon ve [1-40] sınır olarak seçilmiştir. Beyaz ve siyah oyuncuların yöntemleri, yöntemlerden hangisinin kazandığı ve kaç hamlede oyunun bittiği Çizelge 4.18'de verilmiştir.

Çizelge 4.18. Yöntemlerin sıfır oyunda birbirleriyle karşılaştırılması

	Beyaz Oyuncu Yöntemi	**Siyah Oyuncu Yöntemi**	**Oyunun Kazananı**	**Hamle Sayısı**
1	KTGO	YAK	KTGO	69
2	KTGO	YAK	KTGO	46
3	KTGO	YAK	KTGO	41
4	YAK	KTGO	KTGO	42
5	YAK	KTGO	KTGO	49
6	YAK	KTGO	KTGO	42
7	KTGO	PSO	KTGO	89
8	KTGO	PSO	KTGO	69
9	KTGO	PSO	KTGO	56
10	PSO	KTGO	KTGO	72
11	PSO	KTGO	PSO	22
12	PSO	KTGO	KTGO	67
13	KTGO	GA	KTGO	34
14	KTGO	GA	KTGO	32
15	KTGO	GA	GA	54
16	GA	KTGO	KTGO	60
17	GA	KTGO	GA	42
18	GA	KTGO	KTGO	27
19	KTGO	GYA	KTGO	56
20	KTGO	GYA	KTGO	42
21	KTGO	GYA	KTGO	49
22	GYA	KTGO	KTGO	60
23	GYA	KTGO	KTGO	41
24	GYA	KTGO	KTGO	29
25	YAK	PSO	PSO	37
26	YAK	PSO	YAK	29
27	YAK	PSO	YAK	52
28	PSO	YAK	Berabere	80
29	PSO	YAK	PSO	30
30	PSO	YAK	PSO	49
31	YAK	GA	GA	44
32	YAK	GA	GA	64
33	YAK	GA	GA	54
34	GA	YAK	GA	50
35	GA	YAK	YAK	36

Çizelge 4.18. Yöntemlerin sıfır oyunda birbirleriyle karşılaştırılması (Devam)

36	GA	YAK	YAK	84
37	YAK	GYA	GYA	48
38	YAK	GYA	YAK	50
39	YAK	GYA	YAK	26
40	GYA	YAK	GYA	37
41	GYA	YAK	YAK	30
42	GYA	YAK	YAK	35
43	PSO	GA	GA	54
44	PSO	GA	GA	44
45	PSO	GA	PSO	61
46	GA	PSO	PSO	63
47	GA	PSO	GA	62
48	GA	PSO	GA	43
49	PSO	GYA	PSO	91
50	PSO	GYA	GYA	54
51	PSO	GYA	PSO	71
52	GYA	PSO	GYA	40
53	GYA	PSO	PSO	26
54	GYA	PSO	PSO	27
55	GA	GYA	GA	49
56	GA	GYA	GA	47
57	GA	GYA	GA	37
58	GYA	GA	Berabere	83
59	GYA	GA	GYA	26
60	GYA	GA	GA	35

Çizelge 4.18'ten elde edilen sonuçlara göre yöntemlerin 60 oyunda toplam kazandıkları oyun sayısı Çizelge 4.19'da verilmiştir.

Çizelge 4.19. Yöntemlerin sıfır oyunda kazandıkları oyun sayısı

	Toplam Kazanılan Oyun Sayısı
KTGO	21
YAK	8
PSO	10
GA	14
GYA	5

Çizelge 4.19 incelendiğinde KTGO'nun açık ara diğer sezgisel yöntemlerden önde olduğu görülmektedir. GA'nın da iyi performans sergilediği oyunlarda diğer yöntemler başa baş sonuçlar almışlardır.

Sezgisel yöntemlerin birbirleri ile kıyaslanmasında, aralarında yaptıkları maçların yanı sıra hesaplama süreleri de kriter olarak alınabilir. Bunun için yöntemlerin algoritma karmaşıklıkları değerlendirilmelidir. Bu yöntemlerin temel algoritmaları göz önüne alındığında algoritma karmaşıklıkları KTGO, YAK, PSO ve GYA için $O(n^3)$ olurken, GA için $O(n^4)$ olmaktadır (Pençe vd., 2015). Fakat sezgisel yöntemler gibi karmaşık algoritmalarda, algoritma karmaşıklığı birbirine yakın çıktığından her zaman ayırt edici bir kriter olmamaktadır. Bunun yerine bu tarz algoritmaların hızları birim iterasyonu tamamlama süreleri ile daha iyi ölçülebilmektedir.

Algoritma karmaşıklığının daha net hesaplanabilmesi için bu çalışmada 1 iterasyonun tamamlanma süreleri hesaplanıp karşılaştırılmıştır. Algoritmaların birbirlerine karşı süre avantajını karşılaştırmak amacıyla kullanılmaktadır. Her bir algoritmanın 1 iterasyonu gerçekleştirme süreleri hesaplanırken, yöntemler 3 defa çalıştırılıp ortalama süreler dikkate alınmıştır. Zaman değerlerinin tespit edilebilmesi için oyun sonu problemlerinden 2832. oyun baz alınarak hesaplama gerçekleştirilmiştir. Siyah oyuncunun başladığı oyunda siyaha ait sadece Şah yer alırken, beyaza ait ise Kale ve Şah bulunmaktadır. Hem tek taşlı siyah için, hem de iki adet taşa sahip beyaz oyuncu için süre değerleri ayrı ayrı hesaplanmıştır. Standart parametreler olarak popülasyon sayısı 9, iterasyon sayısı 1, sınır değerleri olarak [1-2] alınmıştır. Derinlik bilgisi ise değiştirilip, her bir derinlikteki değerler ayrıca hesaplanmıştır. Algoritmaların birim iterasyon süresi değerleri Çizelge 4.20 ve Çizelge 4.21'de görülmektedir.

Çizelge 4.20. 2832. Oyunda siyah oyuncuya ait sezgisel algoritmaların birim iterasyon süre değerleri

Yöntem	Derinlik/ Populasyon/Sınır	1.Oyun Süre(Sn)	2.Oyun Süre(Sn)	3.Oyun Süre(Sn)	Ortalama Süre (Sn)
KTGO	2/9/[1-2]	0,1416	0,1717	0,1421	0,1518
YAK	2/9/[1-2]	0,1087	0,1229	0,1102	0,1140
PSO	2/9/[1-2]	0,1020	0,1070	0,1054	0,1048
GA	2/9/[1-2]	0,1211	0,1079	0,1658	0,1316
GYA	2/9/[1-2]	0,0936	0,0940	0,1258	0,1045
KTGO	3/9/[1-2]	0,2534	0,2636	0,2543	0,2571
YAK	3/9/[1-2]	0,2229	0,2179	0,2277	0,2228
PSO	3/9/[1-2]	0,2259	0,2088	0,2091	0,2147

Çizelge 4.20. 2832. Oyunda siyah oyuncuya ait sezgisel algoritmaların birim iterasyon süre değerleri (Devam)

GA	3/9/[1-2]	0,2039	0,2767	0,2080	0,2296
GYA	3/9/[1-2]	0,1902	0,2135	0,1946	0,1995
KTGO	4/9/[1-2]	0,3718	0,4130	0,3447	0,3765
YAK	4/9/[1-2]	0,2952	0,2854	0,3096	0,2968
PSO	4/9/[1-2]	0,3044	0,3080	0,2875	0,3000
GA	4/9/[1-2]	0,2840	0,2708	0,2987	0,2846
GYA	4/9/[1-2]	0,2958	0,2698	0,2905	0,2854
KTGO	5/9/[1-2]	0,4627	0,4725	0,4749	0,4701
YAK	5/9/[1-2]	0,3900	0,4050	0,3838	0,3930
PSO	5/9/[1-2]	0,4342	0,3847	0,4492	0,4227
GA	5/9/[1-2]	0,3866	0,3768	0,4032	0,3889
GYA	5/9/[1-2]	0,4617	0,3928	0,3658	0,4068

Çizelge 4.21. 2832. Oyunda beyaz oyuncuya ait sezgisel algoritmaların birim iterasyon süre değerleri

Yöntem	**Derinlik/ Populasyon/Sınır**	**1.Oyun Süre(Sn)**	**2.Oyun Süre(Sn)**	**3.Oyun Süre(Sn)**	**Ortalama Süre (Sn)**
KTGO	2/9/[1-2]	0,1714	0,1745	0,1847	0,1769
YAK	2/9/[1-2]	0,1251	0,1195	0,1207	0,1218
PSO	2/9/[1-2]	0,1203	0,1112	0,1243	0,1186
GA	2/9/[1-2]	0,1556	0,1238	0,1319	0,1372
GYA	2/9/[1-2]	0,1321	0,1262	0,1409	0,1331
KTGO	3/9/[1-2]	0,2777	0,2707	0,2126	0,2537
YAK	3/9/[1-2]	0,2353	0,1891	0,1729	0,1991
PSO	3/9/[1-2]	0,1963	0,2044	0,1858	0,1955
GA	3/9/[1-2]	0,2233	0,2284	0,2283	0,2267
GYA	3/9/[1-2]	0,2041	0,2856	0,2218	0,2372
KTGO	4/9/[1-2]	0,3989	0,3505	0,3343	0,3613
YAK	4/9/[1-2]	0,3186	0,3203	0,3138	0,3176
PSO	4/9/[1-2]	0,3817	0,2455	0,2468	0,2914
GA	4/9/[1-2]	0,3070	0,3520	0,2343	0,2978
GYA	4/9/[1-2]	0,3085	0,2837	0,2896	0,2940
KTGO	5/9/[1-2]	0,4603	0,3710	0,4597	0,4304
YAK	5/9/[1-2]	0,3454	0,4068	0,3733	0,3752
PSO	5/9/[1-2]	0,3865	0,2982	0,3992	0,3613
GA	5/9/[1-2]	0,3639	0,3802	0,3445	0,3629
GYA	5/9/[1-2]	0,2280	0,3610	0,4087	0,3326

Çizelge 4.20 ve 4.21 incelendiğinde derinlik arttıkça amaç fonksiyonunun yapılabilecek doğru hamlelere ait kuralları kontrol etmesi sebebiyle tüm

algoritmalarda birim sürelerin arttığı görülmüştür. Algoritmaların hız değerlerinin yaklaşık aynı olduğu görülmekte olup, aşırı fark değerleri oluşmadığından dolayı yöntemlerin performans üstünlüklerinin doğru hesaplamaya bağlı olarak elde ettikleri puanlarda öne çıktığı tespit edilebilmektedir. Buna rağmen en hızlı yöntem olarak GYA ön plana çıkmakta olup, KTGO'nun ise Tabu'ya benzer eleme yapısının gecikmede etkisi olduğu tespit edilmiştir. Bu rağmen Çizelge 4.19'da da görüldüğü gibi hız açısından KTGO'nun az bir dezavantajı olmasına rağmen elde ettiği açık ara başarılı sonuçlar bu handikaba değdiğini göstermektedir.

5. SONUÇ VE ÖNERİLER

Günümüzde insan ve makine arasındaki en büyük fark idrak etme ve deneyimle kazanılmış bilgileri kullanabilme noktasında ortaya çıkmaktadır. Sadece satranç oyunu için değil, başlangıç koşullarına bağlı bütün durumlarda bu fark oldukça açıktır. Bu çalışmada satranç oyununa farklı bir bakış açısı getirilerek belirli oyun durumları için sezgisel arama işlemi gerçekleştirilmiş ve daha derin olan durum uzayı taranması başarılı bir şekilde gerçekleştirilebilmiştir. Mevcut budama yöntemlerinin derinlerde tıkanma sorununa getirilen sezgisel arama çözümlerine ait literatürde de çok az sayıda çalışma yer almakla birlikte arama işleminin mevcut yöntemlerin parametreleri ile oynama yerine direk oyun ağacında hesaplatıp bulma konusunda satranç oyunu için yeni bir yöntem geliştirilmiştir.

En iyi çözümün yer aldığı uygulamalar sadece oyunlarda değil diğer birçok alanda da yer almaktadır. Öyle ki insan yüzünün 3 boyutlu modellenmesi, mermer, deri ve kumaşların kesilip en optimum şekilde yerleştirilmesi, sınıflandırma algoritmalarının parametrelerinin bulunması gibi önemli alanlarda sezgisel algoritmalardan yararlanılarak arama işlemleri yapılmaktadır.

Bu çalışmada satranç oyununa ait bazı test problemleri üzerinde literatürde yer alan popüler sezgisel algoritmalar yeni geliştirilen arama mekanizması ile birleştirilip, sonuçlar karşılaştırılmıştır. Sezgisel algoritmaların birbirlerine karşı üstünlükleri olduğu gibi, genel olarak hepsi minimaks gibi klasik arama yöntemlerine göre derinlere inildikçe daha hızlı çalışmaktadır. Tüm uzayı taramak yerine umut vadeden bölgelere yoğunlaşan sezgisel yöntemler, her zaman optimum sonucu bulmayı garanti etmese de, deneysel çalışmalarda elde edilen başarılı sonuçlar yöntemin kullanılabilirliğini ve literatüre getirdiği yeni yaklaşımı vurgulamaktadır. Özellikle ünlü satranç şampiyonlarının karşılaşmalarından derlenip, ileriki durumlar düşünülüp, en iyi hamlenin puanlandırıldığı SCHACKNYTT test problemlerinde elde edilen puanlar yöntemin daha da geliştirilmesi için umut vadetmektedir. Geliştirilen yeni

arama algoritması yine bu çalışmada yapılan yeni satranç yazılımına dahil edilerek bir satranç motoru oluşturulmuştur. Oluşturulan satranç motoru oyun bitimine sayılı taş kalan problem örneklerinde de üstün başarı göstermekle birlikte, yazılımın kapanış veritabanı kullanmayıp bellekte de az yer kaplamasını sağlamaktadır. Boyutu terabaytlara kadar gelmiş kapanış veritabanları ile kıyaslanamasa bile, oyun sonunu getirebilecek mat hamlesini bulmada yöntem oldukça başarılıdır. En iyi sonuçların KTGO ile alındığı yazılım, yine en iyi süreleri bu algoritma ile elde etmiştir. Bu sonuçların alınmasında KTGO'nun literatürdeki diğer sezgisel yöntemlere karşı üstünlüğü de etkili olmuştur.

Tüm oyunun oynandığı deneysel çalışmalarda, oyunun başlangıç anında herhangi bir çözümün olmaması sebebiyle yaşanan sezgisel algoritmalardaki dezavantaj açılış kitabı ile giderilmeye çalışılmıştır. Yine de taş adedinin aşırı fazla olduğu durumlarda sezgisel algoritmaların verimi düşmekle birlikte, oyun sonuna yaklaştıkça yöntemin etkisi daha fazla olmaktadır. Bu kategoride de KTGO'nun kazandığı oyun sayısı diğer sezgisel yöntemlere göre oldukça fazla olmuştur.

İleride yapılacak çalışmalarda yöntemin performans ve hız değerlerinin arttırılması için literatürde yer alan ve oyun tahtasını bitlerle ifade eden Bitboard yapısı C++ programlama diliyle kullanılıp hızlandırma işlemi yapılabilir. Ayrıca sezgisel arama yöntemi klasik α-β yöntemi ile birleştirilip melez bir algoritmanın geliştirilmesi planlanmaktadır. Bunlara ek olarak KTGO için kullanılan tabu mekanizması daha da geliştirilerek YSA gibi modellerle gelecek hamle tahmini yardımıyla bu mekanizmanın desteklenmesi planlanmaktadır.

Deneysel çalışma olarak ise ileride, yöntemin ünlü satranç motorlarına karşı test edilmesi amaçlanmaktadır. Sezgisel yöntemler ile arama yapısının Go ve dama oyunu ve benzeri durum uzayı büyük oyunlarda da denenmesi de yapılabilecek çalışmalar arasındadır.

KAYNAKLAR

Akay, B., 2009. Nümerik Optimizasyon Problemlerinde Yapay Arı Kolonisi (Artificial Bee Colony) Algoritmasının Performans Analizi. Erciyes Üniversitesi Fen Bilimleri Enstitüsü, Doktora Tezi, 301s, Kayseri.

Akpınar, F., 2009. Yerleştirme Rotalama Problemi İçin Bir Genetik Algoritma. İstanbul Teknik Üniversitesi Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, 61s, İstanbul.

Al-Khateeb, B., Kendall, G., 2011. The Importance of Look-Ahead Depth in Evolutionary Checkers. In Evolutionary Computation (CEC), 2011 IEEE Congress on, 2252-2258.

Anh, N. D., Nhat, L. T., Nhan, T. V. P., 2016. Design and Control Automatic Chess-Playing Robot Arm. In AETA 2015: Recent Advances in Electrical Engineering and Related Sciences, Springer International Publishing, 485-496.

Arenz, O., 2012. Monte Carlo Chess. Fachgebiet Knowledge Engineering, Lisans Tezi, 35s, Darmstadt.

Arumugam, M.S., Rao, M.V.C., 2008. A New and Improved Version of PSO with Global-Local Best Parameters. Knowledge and Information Systems, 16(3), 331-357.

Bache, K., Lichman, M., 2013. UCI Machine Learning Repository, http://archive.ics.uci.edu/ml. University of California, School of Information and Computer Science.

Beal, D.F., 1990. A Generalised Quiescence Search Algorithm. Artificial Intelligence, 43(1), 85-98.

Björnsson, Y., Marsland, T.A., 2002. Learning Control of Search Extensions. In Proceedings of the 6th Joint Conference on Information Sciences, Durham, 446–449.

Bošković, B., Brest, J., Zamuda, A., Greiner, S., Žumer, V., 2010. History Mechanism Supported Differential Evolution for Chess Evaluation Function Tuning. Soft Computing, 15(4), 667-683.

Campbell, M.S., Marsland, T.A., 1983. A Comparison of Minimax Tree Search Algorithms. Artificial Intelligence, 20(4), 347-367.

Cazenave, T., 2001. Generation of Patterns with External Conditions for the Game of Go. Advances in Computer Games, 9, 275-293.

Çakar, K., 2009. Genetik Algoritmalar Yardımıyla Acil Servis İstasyonu Yerleşiminin Optimizasyonu. Atatürk Üniversitesi, Sosyal Bilimler Enstitüsü, Doktora Tezi, 125s, Erzurum.

David, O.E., van den Herik, H.J., Koppel, M., Netanyahu, N.S., 2014. Genetic Algorithms for Evolving Computer Chess Programs. Evolutionary Computation, IEEE Transactions on, 18(5), 779-789.

David-Tabibi, O., Koppel, M., Netanyahu, N. S., 2010. Genetic Algorithms for Automatic Search Tuning. Icga Journal, 33(2), 67-79.

David-Tabibi, O., Koppel, M., Netanyahu, N.S., 2011. Expert-Driven Genetic Algorithms for Simulating Evaluation Function. Genetic Programming and Evolvable Machines, 12 (1), 5-22.

Duro, J. A., de Oliveira, J. V., 2008. Particle Swarm Optimization Applied to the Chess Game. In IEEE Congress on Evolutionary Computation, 3702-3709.

Edgar, T.F., Himmelblau, D.M., 1988. Optimization of Chemical Processes. McGraw Hill, New York.

Eldem, H., 2014. Karınca Koloni Optimizasyonu (KKO) ve Parçacık Sürü Optimizasyonu (PSO) Algoritmaları Temelli Bir Hiyerarşik Yaklaşım Geliştirilmesi. Selçuk Üniversitesi Fen Bilimleri Enstitüsü, Yüksek Lisans Tezi, 89s, Konya.

Fogel, D. B., Hays, T. J., Hahn, S. L., Quon, J., 2004. A Self-Learning Evolutionary Chess Program. Proceedings of the IEEE, 92(12), 1947-1954.

Forster, B., 2015. Erişim Tarihi: 23.10.2015. http://www.triplehappy.com/.

Gültekin, Ö., 2006. Satranç ve Yapay Zeka Tartışmalarındaki Yeri. Journal of İstanbul Kültür University, 3, 119-128.

Hauptman, A., Sipper, M., 2005. Analyzing the Intelligence of a Genetically Programmed Chess Player. In Late Breaking Paper at Genetic and Evolutionary Computation Conference (GECCO'2005), Washington, DC, USA, 25-29.

Hong, T. P., Huang, K. Y., Lin, W. Y., 2002. Applying Genetic Algorithms to Game Search Trees. Soft Computing, 6(3-4), 277-283.

Iba, W., 2012. Searching for Better Performance on the King-Rook-King Chess Endgame Problem. FLAIRS Conference, 23-25 May, Florida.

Iba, W., Marshman, K., Fisk, B., 2008. Evaluating a Parallel Evolutionary Algorithm on the Chess Endgame Problem. GEM, 235-240.

Iqbal, A., Van der Heijden, H., Guid, M., Makhmali, A., 2012. Evaluating the Aesthetics of Endgame Studies: a Computational Model of Human Aesthetic Perception. Computational Intelligence and AI in Games, IEEE Transactions on, 4(3), 178-191.

Jianhua, W., 1988. The Theory of Games. Oxford University Press.

Juang, C.F., Lu, C.F., 2006. Load-Frequency Control by Hybrid Evolutionary Fuzzy PI Controller. In Generation, Transmission and Distribution, IEEE Proceedings, 153(2), 196-204.

Karaboğa, D., 2005. An Idea Based on Honey Bee Swarm for Numerical Optimization. Technical Report,TR06, Erciyes University Engineering Faculty Computer Engineering Department, Kayseri.

Kennedy, J., Eberhart, R., 1995. Particle Swarm Optimization, Proceedings IEEE International Conference on Neural Networks, Piscataway, 1942-1948.

Knuth, D.E., Moore, R.W., 1975. An Analysis of Alpha-Beta Pruning. Artificial Intelligence, 6, 293-326.

Marsland, T.A., 1986. A Review of Game-Tree Pruning. International Computer Chess Association Journal, 9(1), 3-19.

Moriarty, D. E., Miikkulainen, R., 1994. Evolving Neural Networks to Focus Minimax Search. In Proceedings of the 12th National Conference on Artificial Intelligence, Seattle, 1371-1377.

Myerson, R.B.,1991. Game Theory Analysis of Conflict, Harvard University Press, 7-9.

Nabiyev, V. V., 2012. Yapay Zeka. Seçkin Yayınları, 776s, Ankara.

Pence, I., Cesmeli, M.S., Senel, F.A., Cetisli, B., 2016. A New Unconstrained Global Optimization Method Based on Clustering and Parabolic Approximation. Expert Systems With Applications, 55, 493-507.

Paulsen, P., Fürnkranz, J., 2010. A Moderately Successful Attempt to Train Chess Evaluation Functions of Different Strengths. Proceedings of the ICML-10 Workshop on Machine Learning and Games. Haifa, Israel.

Reinefeld, A., 1983. An Improvement of the Scout Tree-Search Algorithm. Chess Association Journal, 4(6), 4-14.

Sastry, K., Goldberg, D.E., Kendall, G., 2014. Genetic Algorithms. Search Methodologies, Springer US, 93-117.

Seralathan, A. M., 2011. Using Adaptive Learning Algorithms to Make Complex Strategic Decisions. Haverford College, 68p, Pennsylvania.

Shannon, C., 1950. Programming a Computer for Playing Chess. Philosophical Magazine, 7(41), 256-275.

Shi, Y., Eberhart, R., 1998. A Modified Particle Swarm Optimizer. Proceeding of the IEEE International Conferance on Evolutionary Computation, Piscataway, 69-73.

Singh, D., Thaker, C.S., Shah, S.M., 2011. Quality of State Improvisation Through Evaluation Function Optimization in Genetic Application. Emerging Trends in Networks and Computer Communications, 2011 International Conference (IEEE), 22-24 April, 93-97.

Stoiljkovikj, S., Bratko, I., Guid, M., UNI, F., 2015. A Computational Model for Estimating the Difficulty of Chess Problems. In Proceedings of the Third Annual Conference on Advances in Cognitive Systems ACS, 1-16.

Tom Kerrigan's Simple Chess Program (TSCP), 2014. Erişim Tarihi: 23.04.2014. www.tckerrigan.com/Chess/TSCP.

Turing, A.M., Bates, M.A., Bowden, B.V., Strachey, C., 1953. Digital Computers Applied to Games. Faster than Thought, 101.

Vázquez-Fernández, E., Coello, C.A.C., Troncoso, F.D.S., 2012. An Evolutionary Algorithm Coupled with the Hooke-Jeeves Algorithm for Tuning a Chess Evaluation Function. In Evolutionary Computation (CEC), IEEE Congress on, 1-8.

Vázquez-Fernández, E., Coello, C.A.C., Troncoso, F.D.S., 2012. Assessing the Positional Values of Chess Pieces by Tuning Neural Networks' Weights With an Evolutionary Algorithm. In World Automation Congress (WAC), IEEE congress, 1-6.

Vázquez-Fernández, E., Coello, C.A.C. 2013. An Adaptive Evolutionary Algorithm Based on Tactical and Positional Chess Problems to Adjust the Weights of a Chess Engine. In Evolutionary Computation (CEC), IEEE Congress on, 1395-1402.

Vázquez-Fernández, E., Coello, C.A.C., Troncoso, F.D.S., 2013. An Evolutionary Algorithm with a History Mechanism for Tuning a Chess Evaluation Function. Applied Soft Computing, 13(7), 3234-3247.

Vikipedi, 2015. Erişim Tarihi: 23.10.2015. http://tr.wikipedia.org/wiki/Satran%C3%A7.

Von Neumann, J., 1928. Zur Theorie der Gesellschaftsspiele. Math. Annalen. 295-320.

Vuckovic, V., 2015. Candidate Moves Method implementation in MiniMax Search Procedure of the Achilles Chess Engine. In Telecommunication in Modern Satellite, Cable and Broadcasting Services (TELSIKS), 2015 12th International Conference on, 314-317.

Wikipedia, 2015. Erişim Tarihi: 07.04.2015. http://en.wikipedia.org/wiki/Minimax.

Wirth, C., Fürnkranz, J., 2015. On Learning From Game Annotations. Computational Intelligence and AI in Games, IEEE Transactions on, 7(3), 304-316.

Yang, X.S., 2010. A New Metaheuristic Bat-Inspired Algorithm. Nature Inspired Cooperative Strategies for Optimization, Springer Berlin Heidelberg, 65-74.

Yılmaz, S., Küçüksille, E.U., 2015. A New Modification Approach on Bat Algorithm for Solving Optimization Problems. Applied Soft Computing, 28, 259-275.

Yokoyama, S., Kaneko, T., Tanaka, T., 2015. Parameter-Free Tree Style Pipeline in Asynchronous Parallel Game-Tree Search. In Advances in Computer Games, Springer, 210-222.

Zakharov, V. B., Mal'kovskii, M. G., Shchukin, V. Y., 2016. Compression of Underdetermined Data in a 7-piece Chess Table. Moscow University Computational Mathematics and Cybernetics, 40(1), 47-52.

Zaki, M.R., Jaleh, V. and Milad, F., 2015. Preparation of Agar Nanospheres: Comparison of Response Surface and Artificial Neural Network Modeling by a Genetic Algorithm Approach. Carbohydrate polymers, 122, 314-320.

Zhang, J.R., Zhang, J., Lok, T.M., Lyu, M.R., 2007. A Hybrid Particle Swarm Optimization-Back-Propagation Algorithm for Feedforward Neural Network Training. Applied Mathematics and Computation, 185, 1026-1037.

Zhao, F., Ren, Z., Yu, D., Yang, Y., 2005. Application of an Improved Particle Swarm Optimization Algorithm for Neural Network Training. Proceedings of the IEEE International Conference on Neural Networks and Brain, October, Beijing, China, 1693-1698.

ÖZGEÇMİŞ

Adı Soyadı : Melike ŞİŞECİ ÇEŞMELİ

Doğum Yeri ve Yılı : Isparta, 1984

Medeni Hali : Evli

Yabancı Dili : İngilizce

E-posta : ustatmel@hotmail.com

Eğitim Durumu

Lise : Isparta Anadolu Teknik Lisesi, 2002

Lisans : Mersin Üniversitesi, Tarsus Teknik Eğitim Fakültesi, Kontrol Öğretmenliği, 2009

Yüksek Lisans : Süleyman Demirel Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği, 2012

Mesleki Deneyim

Mehmet Akif Ersoy Üniversitesi Ağlasun Meslek Yüksekokulu 2011-2011

Mehmet Akif Ersoy Üniversitesi Bucak ZTYO 2012-.. (halen)

Yayınları

Pence, I., Cesmeli, M.S., Senel, F.A., Cetisli, B., 2016. A New Unconstrained Global OptimizationMethod Based on Clustering and Parabolic Approximation. Expert Systems With Applications, 55, 493-507.

Şişeci, M., Metlek, S., Cetişli, B., 2014. Alt-Bloklar Tekniği ve Kümeleme Yöntemleri ile Görüntü Bölütlemenin Hızlandırılması. Journal of the Faculty of Engineering and Architecture of Gazi University, 29(4), 655-664.

Bozkurt, Ö.Ç., Kalkan, A., Öztop, S., Çeşmeli, M.Ş., 2015. Common Factors in the Swot Analyses of Metropolitan Municipalities in Turkey. Journal of Global Strategic Management, 9(2), 39-49.

Kalkan, A., Bozkurt, Ö.Ç., Öztop, S., Çeşmeli, M.Ş., 2015. Strategic Management Approach in the Metropolitan Municipalities in Turkey: An Analysis on

the Statements of Mission, Vision and Core Values. Journal of Global Strategic Management, 9(1), 62-71.

Şişeci, M., Cetişli, B., 2012. Traverten Plaka Taşlarda Sınıfların K-ortalamalar ve Bulanık C-ortalamalar Kümeleme Yöntemleri ile Belirlenmesi. Süleyman Demirel Üniversitesi, Fen Bilimleri Enstitüsü Dergisi, 16(3), 238-247.

Bozkurt, Ö.Ç., Kalkan, A., Çeşmeli, M.Ş., 2015. Türiye'de Yönetim Bilişim Sistemleri Bölümü Öğrencilerinin Mesleğe Bakış Açıları, Beklenti ve Memnuniyet Düzeyleri. Dokuz Eylül Üniversitesi Yönetim Bilişim Sistemleri Dergisi, 1(2), 22-35.

Çeşmeli, M.Ş., Bozkurt, Ö.Ç., Kalkan, A., Pençe, İ., 2015. Yönetim Bilişim Sistemleri Bölümü Öğrencilerinin Yönetim ve Bilişim Derslerindeki Başarılarının Veri Madenciliği Yöntemleri ile İncelenmesi. Dokuz Eylül Üniversitesi Yönetim Bilişim Sistemleri Dergisi, 1(2), 36-47.

Bozkurt, Ö.Ç., Kalkan, A., Öztop, S., Çeşmeli, M.Ş., 2015. Common Factors in the Swot Analyses of Metropolitan Municipalities in Turkey. 11th International Strategic Management Conference, July 23-25, Vienna, 77-85.

Kalkan, A., Bozkurt, Ö.Ç., Öztop, S., Çeşmeli, M.Ş., 2015. Strategic Management Approach in the Metropolitan Municipalities in Turkey: An Analysis on the Statements of Mission, Vision and Core Values. 11th International Strategic Management Conference, July 23-25, Vienna, 67-76.

Uğuz, S., Kılıç, B., Şişeci, M., 2013. Akıllı Ev Otomasyonu Sistemlerinde Zigbee Tabanlı Ağ Uygulamaları. III. Elektrik Tesisat Ulusal Kongresi Kapsamında 6. Kontrol Otomasyon ve Yapı Elektronik Sistemleri Sempozyumu, 21-24 Kasım, İzmir.

Çeşmeli, M., Bozkurt, Ö. Ç., Kalkan, A., 2015. Türkiye'de Otomobil Talebinin Yapay Sinir Ağları ile Tahmin Edilmesi. II. Ulusal Yönetim Bilişim sistemleri Kongresi, 8-10 Ekim, Erzurum, 669-680.

Pençe, İ., Çeşmeli, M.Ş., Aslan, G., 2015. Bulanık Saılarda Dört İşlemi ve İki Bulanık Sayı Arasındaki Uzaklığı Hesaplayan Görsel Ders. II. Ulusal Yönetim Bilişim sistemleri Kongresi, 8-10 Ekim, Erzurum, 790-802.

Bozkurt, Ö.Ç., Kalkan, A., Çeşmeli, M.Ş., 2014. Türiye'de Yönetim Bilişim Sistemleri Bölümü Öğrencilerinin Mesleğe Bakış Açıları, Beklenti ve Memnuniyet Düzeyleri. I. Yönetim Bilişim Sistemleri Kongresi, 16-17 Ekim, Bağaziçi Üniversitesi İstanbul, 18.

Çeşmeli, M.Ş., Bozkurt, Ö.Ç., Kalkan, A., Pençe, İ., 2015. Yönetim Bilişim Sistemleri Bölümü Öğrencilerinin Yönetim ve Bilişim Derslerindeki Başarılarının Veri Madenciliği Yöntemleri ile İncelenmesi. I. Yönetim

Bilişim Sistemleri Kongresi, 16-17 Ekim, Bağaziçi Üniversitesi İstanbul, 17.