ANKARA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

BİLGİSAYAR OYUNLARINDA MAKİNE ÖĞRENMESİ YÖNTEMLERİNİN KULLANILMASI

Hakan Oğuz VURAL

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

ANKARA
2022

ÖZET

Yüksek Lisans Tezi

BİLGİSAYAR OYUNLARINDA MAKİNE ÖĞRENMESİ YÖNTEMLERİNİN KULLANILMASI

Hakan Oğuz VURAL

Ankara Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Prof. Dr. İman ASKERBEYLİ

Bu projede evrimsel yapay sinir ağları teknikleri kullanılarak bir bilgisayar oyununu oynamayı sıfırdan, kendi kendine öğrenen bir yapay zekâ geliştirmek amacıyla Yapay Oyuncu Geliştirme Ortamı (YOGO) adı verilen bir sistem oluşturulmuştur. YOGO sayesinde bilgisayar oyunlarını oynayabilen birleşimli bir yapay oyuncu geliştirilmiş ve statik yapay zekalardan daha yüksek performanslı yapay zekalar oluşturmak amaçlanmıştır. Flappy Bird, Google Dino Run ve Cell Rush adlı üç oyunda YOGA sınanmış ardından birtakım testler yardımıyla bu oyunları istenilen seviyede oynayabilecek, başarı kriterlerini sağlayan yapay oyuncular geliştirilmiştir. Her bir yapay oyuncu, oyun dünyasını algılayabilecek sensörlere, oyun dünyası ile etkileşime girecek hareket motor sistemine ve sensör ile motor sistemi arasında veri akışını sağlayacak bir yapay sinir ağına sahiptir. Yapay oyuncuların sahip olduğu yapay sinir ağları sensörden gelen verileri girdi olarak alıp, işleyip ve çıktı üretmektedir. Üretilen çıktı verileri motor sistemine yollanmakta ve motor sistemi aldığı verilere göre yapay oyuncunun hareketlerini gerçekleştirmektedir. Yapay oyuncular performansına göre değerlendirilerek ve genetik algoritma teknikleriyle seçilip, çaprazlanıp, mutasyona uğrayarak bir sonraki jenerasyona aktarılmaktadır. Sadece en iyi yüzde elli çaprazlamalara katılmaktadır. Çaprazlanan bireyler yeni jenerasyonun bir önceki jenerasyonlardan daha iyi olması ümidiyle seçilmektedir. Yapay oyuncular başarı kriteri sağlanana kadar eğitilmekte ve nesilden nesille aktarılmaktadır.

Temmuz 2022, 56 Sayfa

Anahtar Kelimeler: Makine öğrenmesi, Yapay sinir ağları, Evrimsel sinir ağlar, Yapay zeka

ABSTRACT

Ph. D. Thesis

USING MACHINE LEARNING TECHNIQUES IN COMPUTER GAMES

Hakan Oğuz VURAL

Ankara University
Gradute School of Natural and Applied Sciences
Department of Computer Engineering

Supervisor: Prof. Dr. İman ASKERBEYLİ

In this project, a system called Artificial Player Development Environment (YOGO) was created in order to develop a self-learning artificial intelligence from scratch using evolutionary artificial neural networks techniques. Thanks to YOGO, a composite artificial player that can play computer games has been developed and it is aimed to create higher performance artificial intelligences than static artificial intelligences. YOGA has been tested in three games called Flappy Bird, Google Dino Run and Cell Rush. As a result of the tests, artificial players have been developed that can play these games at the desired level and meet the success criteria. Each artificial player has sensors that can detect the game world, a motion engine system that will interact with the game world and an artificial neural network that will provide data flow between the sensor and the engine system. Artificial neural networks owned by artificial players take the data from the sensor as input, process it and send the output data it produces to the motor system. The motor system performs the movements of the artificial player according to the data it receives. Artificial players are evaluated according to their performance and are selected, crossed, mutated and passed on to the next generation with genetic algorithm techniques. Only the best fifty percent individuals participate in crossing proccess. Crossed individuals are selected in the hope that the new generation will be better than the previous generations. Artificial players are trained and handed down from generation to generation until the success criterion is met.

July 2022, 56 Page

Key Words: Machine learning, Artificial Neural networks, Neuroevolutional networks, Artficial intelligence

ÖNSÖZ VE TEŞEKKÜR

Çalışmalarımda bana yardımcı olan ve yüksek lisans çalışmalarıma devam etmemde bana destek olarak gelişmeme yardımcı olan danışman hocam Sayın Prof. Dr. İman ASKERBEYLİ'ye, çalışmalarım süresince destekte bulunan Sayın Doç. Dr. Mehmet Serdar GÜZEL'e, çalışmalarım esnasında birçok fedakarlık göstererek beni destekleyen eşim ve aileme en içten dileklerimle teşekkür ederim.

Hakan Oğuz VURAL
Ankara, Temmuz 2022

İÇİNDEKİLER

SİMGELER DİZİNİ

ß	Nguyen Widrow Faktörü
n	Girdi Katmanı Düğüm Sayısı
p	Gizli Katmanlardaki Toplam Düğüm Sayısı
v_i	i. Sıradaki Ağırlık Değeri
I	Genç Nüfusun Toplam Nüfusa Oranı
R	Daire Alanın Çapı

Kısaltmalar

YOGO	Yapay Oyun Geliştirme Ortamı
YZ	Yapay Zeka
YO	Yapay Oyuncu

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

1. GİRİŞ

Oyun Teknolojilerinin gelişmesiyle birlikte bilgisayar oyunlarında, insan gibi davranan ve oyuncuya mücadele dolu bir tecrübe yaşatması gereken yapay zekalara ihtiyaç artmıştır. Bu tür yapay zekaların gerçekçi ve mantıklı davranması gerekmektedir. Bu yüzden oyun yapay zekalarını gerçekleştirmek adına son yıllarda birçok yeni yöntem ortaya çıkmıştır (The Verge, 2019). Bilgisayar oyunlarındaki yapay zekalar için kullanılan yöntemlerden birisi de makine öğrenmesidir.

Yapay zekâ teknolojileri tıp, mühendislik, matematik gibi birçok alanda kullanılmaktadır (Burns E. vd., 2022). Yapay zekanın kullanıldığı en eski alanlardan biri de oyun teknolojisidir. Bilgisayar oyunlarında yapay zekanın gerçekçi, mantıklı ve ilgi çekici olması gerekmektedir. Bu gereksinimler oyuncunun oyuna bağlanması ve oyunu beğenmesi için şarttır. Oyun yapay zekasını gerçekçi ve mantıklı yapmak için birçok yöntem geliştirilmiştir ve gün geçtikçe daha fazla yeni yöntem geliştirilmektedir. Önceden kullanılan yöntemlerde yapay zekâ alışılmışın dışına çıkamamaktaydı ve belirli bir süre sonra hareketleri oyuncu tarafından tahmin edilebilir hale gelmekteydi. Bu durum oyunun eğlenceli yanını köreltmektedir ve oyunun popülaritesinin beklenenden daha kısa sürede düşmesine sebep olmaktadır. Yapay zekâ geliştiricilerinin ön göremediği bir takım senaryo ve hareketlerden dolayı çoğu oyun yapay zekâsı oyuncular açısından niteliksiz görülmektedir. Bu problemi çözmek için geliştiriciler kendi kendine öğrenen yapay zekâ modellerini kullanmaya başlamıştır. Yapay zekâ oyun ortamında kendi tecrübelerine göre hareket ettiği için daha gerçekçi ve ilginç kararlar almaktadır ve ayrıca geliştiricilerin ön göremediği senaryolara uygun hareketler oluşturmaktadır. Bu tez çalışmasında en başta hiçbir şey bilmeyen bir yapay zekayı vakit geçtikçe oyunu ortalama bir oyuncu kadar iyi oynayacak seviyeye getirmek için makine öğrenmesi yöntemlerini kullanmak amaçlanmıştır.

2. PROBLEMİN TANIMI

Problem, makine öğrenmesi yöntemleriyle başta hiçbir bilgisi olmadan bir oyunu oynayıp zamanla kendini geliştirebilecek bir yapay zekâ sistemi üretmektir. Bu sistem herhangi bir oyuna uyarlanabilecek şekilde değişken bir sistem olmalıdır. Bu yüzden birden fazla türde oyuna uyumlu çalışabilecek bir tür yapay zekâ sisteminin oluşturulması gerekir. Bu sistem Yapay Oyuncu Geliştirme Ortamı (YOGO) olarak adlandırılmıştır ve sistemin evrimsel yapay sinir ağları tekniklerini kullanarak oluşturulması hedeflenmiştir.

Yapay oyuncu , hedef oyunu oynayarak kendini eğiten ve zamanla gerçek bir oyuncu gibi oyunu oynayabilmesi hedeflenen yapay zekadır. YOGO'yu kullanarak oyun dünyası ile etkileşime girer ve hareket üretir. Üretilen hareketin sonucu uygunluk puanını etkiler. Uygunluk puanı yapay oyuncunun ne kadar iyi oyunu oynayabildiğini ölçen değerdir.

Eğer yapay oyuncu, hedef oyunu ortalama bir oyuncudan daha iyi oynayabiliyorsa istenilen hedefe ulaşmış demektir. Bunun için oyuncaların oyunu oynarken aldığı skorun ortalaması ile yapay oyuncu için hesaplanan uygunluk puanı karşılaştırılacak ve önceden belirlenen skor düzeyini geçene kadar veya sınır iterasyona ulaşana kadar yapay oyuncu eğitilecektir.

3. KURAMSAL TEMELLER

Bu problemin çözülmesinde kullanılan evrimsel yapay sinir ağları oyunlar için yapay zekâ geliştirmede sık tercih edilen popüler bir yöntemdir. Yaklaşık yirmi yıldır kullanılan evrimsel yapay sinir ağlar oyunlara, bir insan gibi hareket eden yapay zekalar tasarlamada etkin bir yöntemdir (Risi S. ve Togelius J., 2017). Evrimsel yapay sinir ağlar ayrıca müzik besteleme (Hoover A. K. vd., 2012), biyolojik fenomen modelleme (Clune J. vd., 2013), çip kaynak paylaştırma (Gomez F. J. vd., 2001), robot kontrolü (Nolfi S. ve Floreano F., 2000) vb. gibi çok sayıda problem için kullanılmış bir yöntemdir.

3.1 Evrimsel Yapay Sinir Ağları

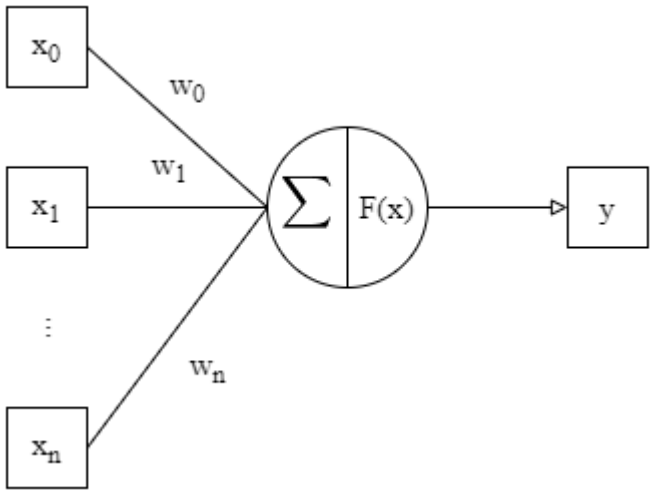
Evrimsel yapay sinir ağları, kısaca genetik algoritmalar ile yapay sinir ağları yöntemlerinin birleşiminden oluşan bir yöntemdir. Yapay zekanın beyni işlevini gören bir yapay sinir ağının nöronları arasındaki ağırlıklar o yapay zekanın genotipini oluşturur. Birden çok yapay zeka test ortamında sınanarak derecelendirilir. Derecelendirme sonucunda kötü performans gösterilen bireyler yok edilirken iyi performans gösteren bireyler tekrar teste sokulmak için ve yeni bireyler oluşturmak için seçilirler. Seçilmiş olan bireyler yeni jenerasyon oluşturmak için çaprazlama ve mutasyon yöntemlerinden geçerler. Bireyler istenilen performansa ulaşana kadar bu işlemler tekrarlanır.

3.1.1 Yapay sinir ağları

Son elli yılda yapay sinir ağları mühendislik bilimi alanında birçok yerde kullanılmıştır (Floerano D. vd., 2008). Sadece mühendislik alanında değil, aynı zamanda beynin yapısını öğrenmede de kullanılan yapay sinir ağları makine öğrenmesi tekniklerinin gelişmesinde önemli rol oynamıştır.

Yapay sinir ağı, beyin modeli ele alınarak oluşturulmuş bir sistemdir. Nöron denilen düğümler ve bu düğümlerin arasındaki bağlantılardan oluşur Nöron (Şekil 3.1.). Birden fazla bağlantıyı girdi olarak alan, bu girdileri toplayıp aktivasyon fonksiyonundan geçirerek tek bir çıktı üreten düğümdür. Bağlantıların her birinin ağırlık değeri vardır. Bu değer o bağlantıdan gelen verinin etkinlik derecesini belirler. Şekil 3.1.'deki nörona gelen n adet bağlantı için n adet ağırlık ([w_0, w_1, w_2, … ,w_{n-1}]) ve n adet girdi ([x_0, x_1, x_2, … ,x_{n-1}]) bulunmaktadır. Bu nöronun aktivasyon fonksiyonu F (x) ise nöronun çıktısı şu şekilde olacaktır:

$$y = F\left(\sum_{i=0}^{n-1} x_i * w_i\right)$$



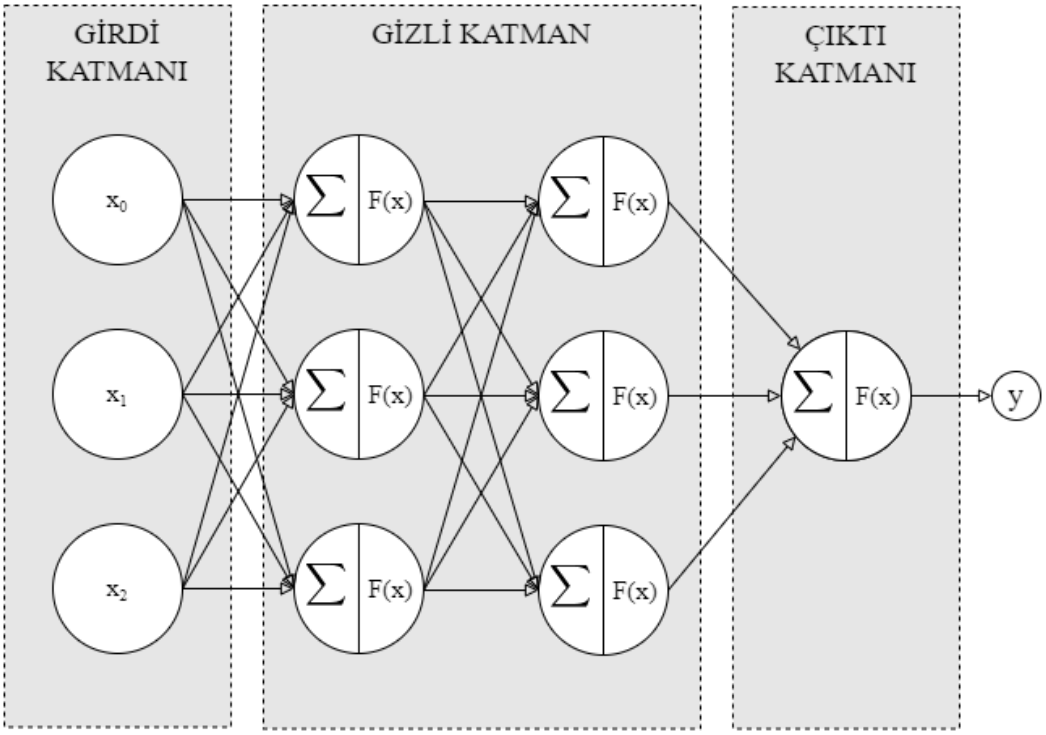
Şekil 3.1 Yapay sinir ağlarında bir nöron yapısı

Yapay sinir ağları katmanlardan oluşur. Her bir katmanda belirli bir sayıda nöron bulunacaktır. Bir nöron kendinden önceki katmandaki tüm nöronlar ile bağlantılıdır. Bu bağlantı önceki nöronların çıktı değerini ve bir ağırlık değeri taşır. İlk katman girdi katmanıdır ve buradaki nöronların sadece çıktı değerleri vardır. Son katman ise çıktı katmanıdır. Girdi katmanı ve çıktı katmanı arasında kalan tüm katmanlar gizli katman olarak adlandırılır.

Örneğin Şekil 3.2.'de gösterilen yapay sinir ağı ikisi gizli katman olmak üzere toplam dört katmana sahip bir yapay sinir ağıdır. Girdi katmanında üç adet nöron, gizli

katmanlarda üçer adet nöron ve son olarak çıktı katmanında bir adet nöron bulunmaktadır. Toplam bağlantı sayısı 21 adet (3x3 + 3x3 + 3x1 = 19) olacaktır. Soldan sağa ve yukarıdan aşağıya tüm ağırlıkları sıraladığımızda 19 elemanlı bir ağırlık dizisi olan W dizisini elde etmiş oluruz:

$$W = [\ w_0, w_1, w_2, \ldots w_{18}]$$



Şekil 3.2 Yapay sinir ağlarına bir örnek

3.1.2 Genotip ve genler

Bir yapay sinir ağının ağırlıklarının dizisi bireyin genotipini oluşturur. Gentopin her bir elemanı gen olacaktır. Genotipdeki veriler başka bireylere aktarılabilir ve bir bireyin yapay sinir ağı oluşturulurken kullanılabilir veya tam tersi bir yapay sinir ağı genotip oluşturabilir.

Evrimsel yapay sinir ağlarında her bir bireyin yapay sinir ağı aynı yapıda olacağı için genotiplerinin uzunluğu da aynı olacaktır. Eşit uzunluktaki bu genotipler çaprazlanarak yeni bireyleri oluşturacaktır.

3.1.3 Popülasyon ve jenerasyonlar

Birden fazla bireyin oluşturduğu küme popülasyonu oluşturur. Popülasyonun içerisindeki her bir bireyin kendine ait yapay sinir ağı ve genotipi bulunur. Popülasyondaki bireyler sınanıp performansına göre sıralandıktan sonra iyi performans gösterenler çaprazlanıp yeni bireyleri oluşturacak ve bu yeni bireylerle birlikte yeni bir popülasyon oluşturacaklardır. Bu iki farklı popülasyon farklı jenerasyonları oluşturur. Tüm jenerasyonlar eşit sayıda bireye sahip olmalıdır. Bu yüzden bir önceki popülasyondaki kötü performanslı bireylerin elenmesi gerekir. Bir jenerasyonun ortalama performansı istenilen değere gelene kadar jenerasyonlar sınanır, yeni jenerasyon üretilir ve böylece devam eder. En başta oluşturulan jenerasyonun genotipleri rastgele oluşturulacak veya ilk jenerasyonun genotiplerinin tüm elemanları sabit bir sayıya eşitlenecektir (genellikle sıfır tercih edilmektedir.) Her ne olursa olsun ağırlık değerleri -1 ile 1 arasında değer alacağı için rastgele oluşturulan genotiplerin değerleri de -1 ile 1 arasında olmalıdır. İlk oluşturulan jenerasyona sıfır jenerasyonu da denmektedir.

3.1.4 Uygunluk fonksiyonu ve adayların seçilmesi

Bir jenerasyondaki bireylerin sınandıktan sonra performanslarını ölçmek için uygunluk fonksiyonu kullanılır. Uygunluk fonksiyonu her bir bireye uygunluk değeri atar. Uygunluk fonksiyonunun içeriği probleme göre değişir. Yine de tüm uygunluk fonksiyonları bireyin problemi çözmeye ne kadar yaklaştığı düşünülerek oluşturulur. Eğer bir birey, yaptığı hareketler ve aldığı kararlarla problemi çözmeye daha fazla yaklaşıyorsa daha çok uygunluk değeri alması beklenir.

Bir jenerasyon sınanıp da tüm bireylere uygunluk değeri atandıktan sonra bireyler en çok uygunluk değeri alandan en az alana göre sıralanır. Sıralanan bireylerden en iyi yüzde ellisi çaprazlanmak ve bir sonraki jenerasyona geçmek için seçilir. Bazı durumlarda geliştirici sonraki jenerasyona geçecek bireyleri daha rastgele hale getirmek için ilk yüzde ellideki rastgele birkaç birey ile son yüzde ellideki birkaç bireyin yerini değiştirmeyi seçebilir. Böylece ilk yüzde elliye girmeyen birkaç "şanslı" birey sonraki jenerasyona geçmiş olur ve çeşitlilik sağlanmış olur. Ayrıca o jenerasyonda kötü performans gösterip de bir sonraki jenerasyonlarda özgün çözümler üretebilecek bireyler de elenmemiş olur. Tabi bu yaklaşım çok rastgele olduğu için istenmeyen sonuçlar da doğurabilir.

Seçilen ilk yüzde elli birey sıradaki jenerasyonun yarısını oluşturur. Sıradaki jenerasyonun diğer yarısı da seçilen bireylerin çaprazlanmasından oluşur. Böylece sonraki jenerasyonun önceki jenerasyonlardan daha iyi olması beklenir. Ancak bazı başlangıç durumlarında sıfır jenerasyonundaki tüm bireyler birbirine yakın ve istenilenden çok uzak bir performans gösterebilmektedir. Bu durumda kötü bireylerin çaprazlanmasından oluşan yeni jenerasyon da bir önceki kadar kötü olacaktır ve ilerleme sağlanamayacaktır. Bu durumu çözmek için iki yol vardır. Birincisi sıfır jenerasyonu birbirine çok yakın ve çok kötü performans gösterdiğinde yeni bir sıfır jenerasyonu başlatılarak her şey başa alınır ve daha iyi bir jenerasyon çıkması ümit edilir. İkinci yöntem ise mutasyondur. Mutasyon kullanılmasındaki amaç çok yavaş gelişim gösteren bireylerin genotipinde rastgele bir değişiklik yapıp o bireylerden daha iyi bireyler oluşturmaya çalışmaktır. Yani sıfır jenerasyonu çok kötü bir başlangıç yaptıysa veya o anki jenerasyon ortalama performansta ilerleme kat etmiyorsa mutasyon oranı arttırılıp bireyleri çeşitlendirmeye gidilebilir.

Bir jenerasyondaki çaprazlama için seçilmiş bireylerin (ilk yüzde elliye giren bireyler) çaprazlamaya gitme sıklıkları için de farklı yollar izlenebilir. Çaprazlanacak bireylerden en yüksek uygunluk değerine sahip birey başta olacak şekilde sıralanmış bir liste oluşturulur. Bu listeden seçilecek iki birey çaprazlanarak yeni bireyi/bireyleri (çaprazlama biçimine bağlı) oluşturacaktır.

Bir bireyin kendi ile çaprazlanması demek o bireyin genotipinin kopyalanıp sonraki jenerasyona geçmesi demektir (eğer hiç mutasyon geçirmemişse). Mutasyon oranı yüksek olan durumlarda bunun sakıncası olmayabilir ancak mutasyon oranının çok düşük olduğu durumlarda bireyin kopyalanması zamanla tüm bireylerin aynı olmasına, jenerasyonun belirli bir performans düzeyinde takılı kalmasına yani problemin çözümünde lokal minimum, lokal maksimum noktasına takınılmasına sebep olacaktır.

Bir bireyin birden fazla çaprazlanması bir sonraki jenerasyonda o bireye benzer birçok bireyin ortaya çıkmasına sebep olur. Eğer bir birey çok iyi performans gösterdiyse bu bireyin bir sonraki jenerasyona geçmesi kısa vadede sorundan çok fayda sağlayacaktır ama uzun vadede tüm bireylerin çok benzer şekilde davranmasına sebep olacağı için çeşitliliği öldürecektir. Eğer bulunduğu jenerasyonun en iyisi olan bir bireyin bir kere çaprazlanmasına izin verilirse jenerasyonların sonuca ulaşma hızı büyük oranda düşecektir. Sonuçta evrimsel yapay sinir ağları yönteminde kullanılan genetik algoritmalar en iyinin neslini devam ettirmesi, en kötülerin de yok olması ve böylece istenilen sonuca ilerlenmesi prensibine dayanır. Bu yüzden uygunluk değeri yüksek bireylerin daha çok çaprazlanması nihayetinde jenerasyonun ortalama performansını yükseltecek bir hamledir.

Eğer bireyler uygunluk değeri ile direk orantılı olarak çaprazlanma şansına sahip olursa ve çaprazlanan bir birey tekrar çaprazlanabilirse zamanla tüm popülasyon en iyi bireye benzeyecek ve çeşitlilik azaldığından problemin çözümünün lokal minimum/maksimum noktasına takılma riski artacaktır. Hem çözüm hızını yüksek tutarak hem de çeşitliliği koruyarak ilerlemenin yolu *sıra tabanlı* çaprazlama şansı atamaktan geçmektedir. Yani çaprazlama şansı bireyin uygunluk değeri ile doğrudan orantılı değildir.

İlk yüzde elliye giren bireylerden hangisinin çaprazlanmaya gireceğini belirlemek için en yaygın kullanılan seçilim şekilleri şunlardır (Holland J. H., 1992):

1. Rulet Tekerleği Seçimi
2. Sıralı Seçim
3. Turnuva Seçimi.

Rulet Tekerleği seçiminde bir bireyin seçilme olasılığı seçime katılacak olan tüm bireylerin uygunluk değerlerinin toplamının o bireyin uygunluk değerine bölümü ile bulunur. Yani bireyin uygunluk değeri diğerlerinden ne kadar yükse ise o kadar çok seçilir. Bu seçim tipi eğer birinci birey ile ikinci birey arasında çok fark varsa popülasyondaki bireylerin jenerasyon geçtikçe birinci bireye benzemesine ve çeşitliliğin azalmasına neden olur.

Sıralı seçimde, en sonuncu bireye (seçilenler arasında) 1 sıralama puanı verilir. Bir kademe daha iyi bireye 2, bir sonrakine 3 vb. alacak şekilde puanlar birer arttırılarak sıralama puanları atanır. Böylece en çok uygunluk puanı alan birey en yüksek sıralama puanına sahip olur. Ancak birinci ve ikinci birey arasındaki sıralama puanı, uygunluk puanı kadar fazla olmayacak ve uygunluk puanı ne olursa olsun sıralama puanı sadece bir puan olacaktır. Rulet tekerleği seçim yöntemindeki hesaplandığına benzer şekilde bir bireyin seçilme olasılığı kendi sıralama puanının tüm bireylerin sıralama puanı toplamına oranı olacaktır. Bu yöntemle, Rulet Tekerleği seçimindeki gibi tüm popülasyonun tek tipe dönmesi engellenmiş olsa da çözüme ulaşma hızı yavaşlar.

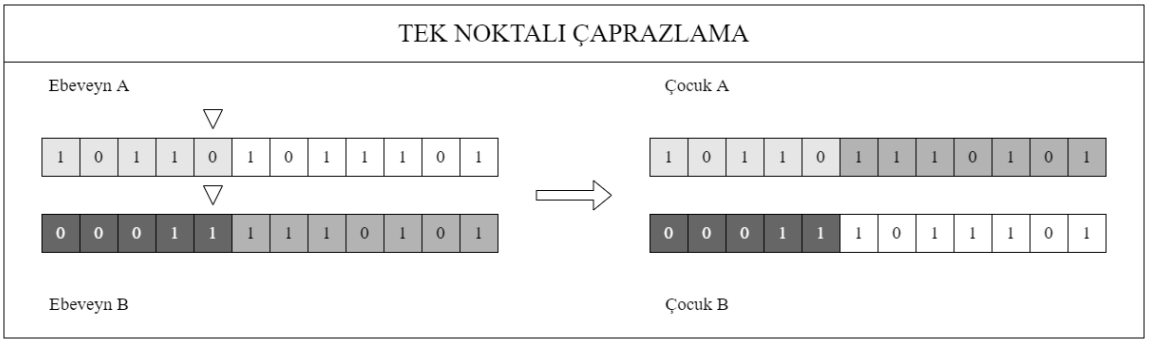
Turnuva seçiminde bir k sayısı belirlenir. Tüm bireylerin arasından k adet birey rastgele seçilir. Seçilen bireyler arasından en yüksek uygunluk değerine sahip birey çaprazlanmak için seçilir. Böylece çeşitlilik ve rastgelelik çok fazla ancak çözüme ulaşma hızı tahmin edilemez olur.

3.1.5 Çaprazlama

İki bireyin genotipi çaprazlanarak yeni birey/bireyler üretilir. Birçok çaprazlama türü vardır ve bu çaprazlama türleri genotipin yapısına ve haliyle probleme bağlı olarak seçilmektedir. Çaprazlamaya uğrayan bireylere ebeveyn, yeni oluşan birey/bireylere çocuk denilmektedir. Çaprazlamada en yaygın kullanılan yöntemler şöyledir (Eren ve Avuçlu 2019):

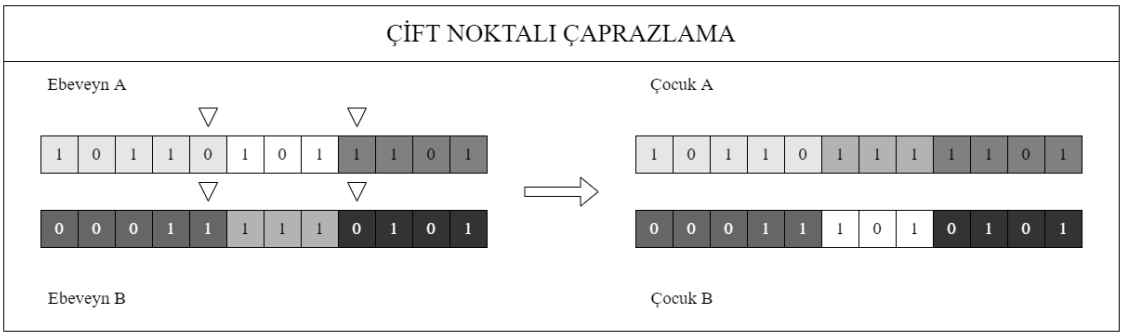
1. Tek noktalı çaprazlama
2. Çift noktalı çaprazlama
3. Sıralı çaprazlama
4. Kes-ekle çaprazlama
5. Tekdüze (uniform) çaprazlama

Tek noktalı çaprazlamada genotip üzerindeki değerlerden biri seçilir. Birinci çocuk birinci ebeveynden seçilen değerin öncesini, ikinci ebeveynden de sonrasını alır. İkinci çocuk ise tam tersi şekilde değerleri alır (Şekil 3.3).



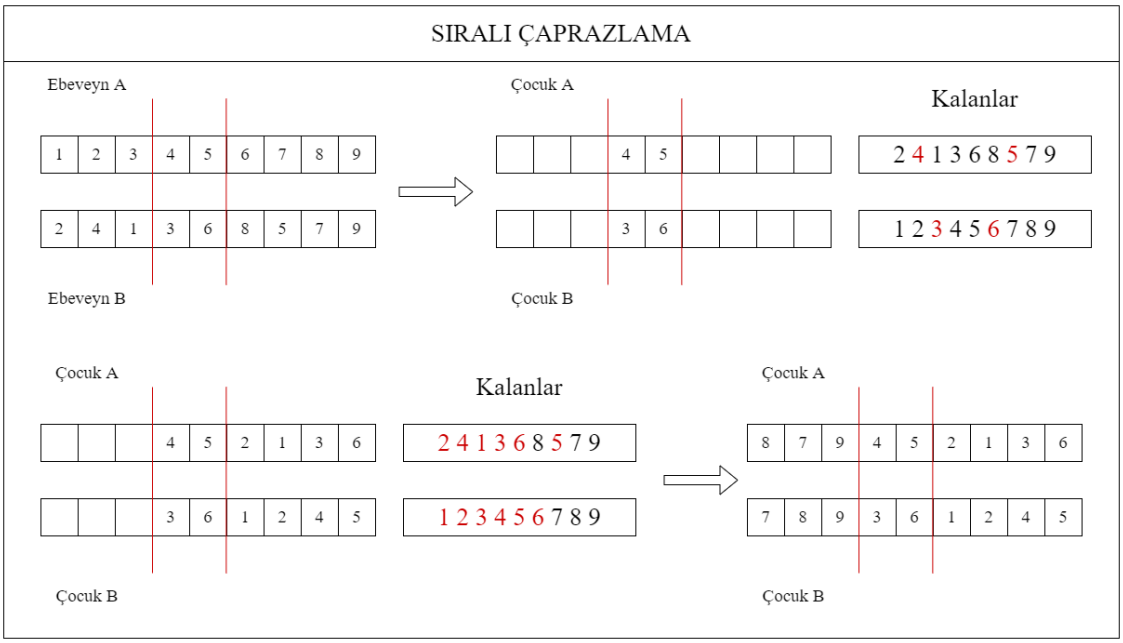
Şekil 3.3 Tek noktalı çaprazlama yöntemi

Çift noktalı çaprazlamada da aynı tek noktalı çaprazlamadaki gibi bir yöntem kullanılır. Ancak bu sefer bir nokta yerine iki nokta seçilir. Bu iki nokta ebeveynlerin genotipini üç parçaya ayırır. İlk çocuk, ebeveyn 1'den birinci ve üçüncü parçayı ebeveyn 2'den ikinci parçayı alır. İkinci çocuk ise tam tersi parçaları alır (Şekil 3.4.).



Şekil 3.4 Çift noktalı çaprazlama yöntemi

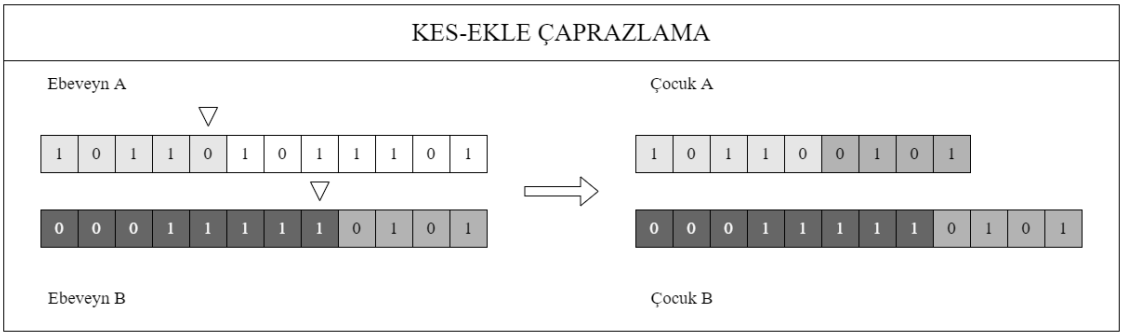
Sıralı çaprazlama genotip üzerindeki değerlerin sırası önemli olduğu durumlarda kullanılır. İlk önce iki nokta belirlenir. Bu noktalar arasında kalan değerler çocuklara olduğu gibi aktarılır. Bu değerlerden sonraki ve önceki yerler şu an için çocuklarda boştur. Çocuk bir, ebeveyn ikideki genlerden kendisinde olmayanları sırasıyla kendi genlerine kopyalar. Aynı işlemi ikinci çocuk, birinci ebeveyn için yapar (Şekil 3.5).



Şekil 3.5 Çift noktalı çaprazlama yöntemi

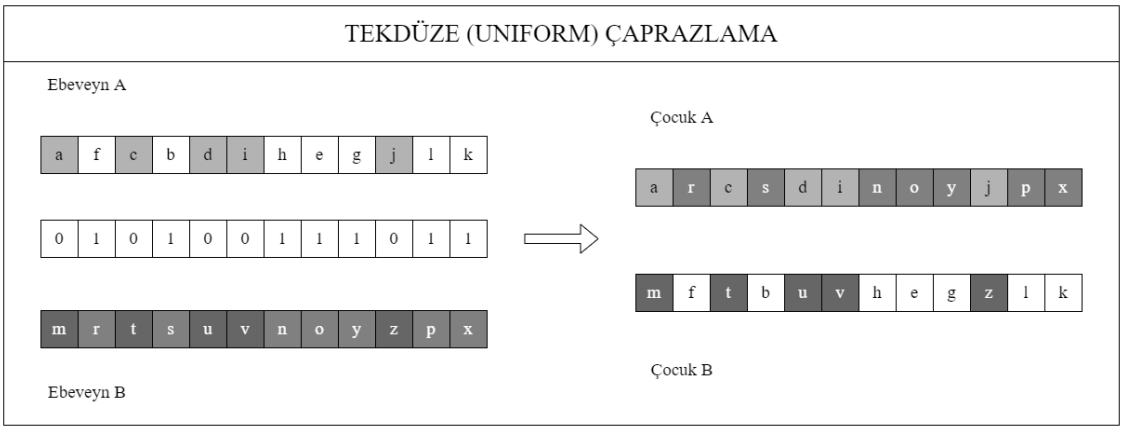
Kes-ekle çaprazlama yönteminde her iki ebeveyn için farklı konumda birer nokta belirlenir. Bu noktalar genotipleri iki parçaya ayırır. İlk çocuk için birinci ebeveynin ilk

parçası ikinci ebeveynin ikinci parçası alınır. İkinci çocuk için de tam tersi alınır. Ebeveynlerin genotipi aynı uzunlukta olsa bile çocukların genotipi farklı uzunluklarda olabilir ve ayrıca genotip uzunluğu farklı olan bireyler de çaprazlama yapabilir (Şekil 3.6).



Şekil 3.6 Kes-ekle çaprazlama yöntemi

Tekdüze (uniform) çaprazlama yönteminde eşit uzunluktaki iki genotip yine aynı uzunluktaki 0-1'lerden oluşan, kalıp adı verilen bir dizi ile karşılaştırılarak çaprazlanır. İlk çocuk, ebeveyn birin kalıp dizisindeki 0 değerine karşılık gelen genler ile aynı değeri alır. İlk çocuğun geri kalan genleri, ikinci ebeveynin kalıp dizisinde 1'e karşılık gelen genleriyle aynı değerde olacaktır. İkinci çocuk için de tam tersi işlemler yapılarak genotipi oluşturulur (Şekil 3.7).



Şekil 3.7 Tekdüze (uniform) çaprazlama yöntemi

3.1.6 Mutasyon

Mutasyon, oluşturulan çocuk bireylerin genlerinin birkaçının, mutasyon oranına göre, rastgele değiştirilmesidir. Böylece popülasyonda çeşitlilik oluşturulmuş olur. Eğer mutasyon oranı yüksek tutulursa bireyin gen yapısının tamamının değişmesi söz konusudur. Eğer mutasyon oranı düşük tutulursa birey tamamen ebeveynlerine benzer ve değişiklik ortadan kalkmış olur.

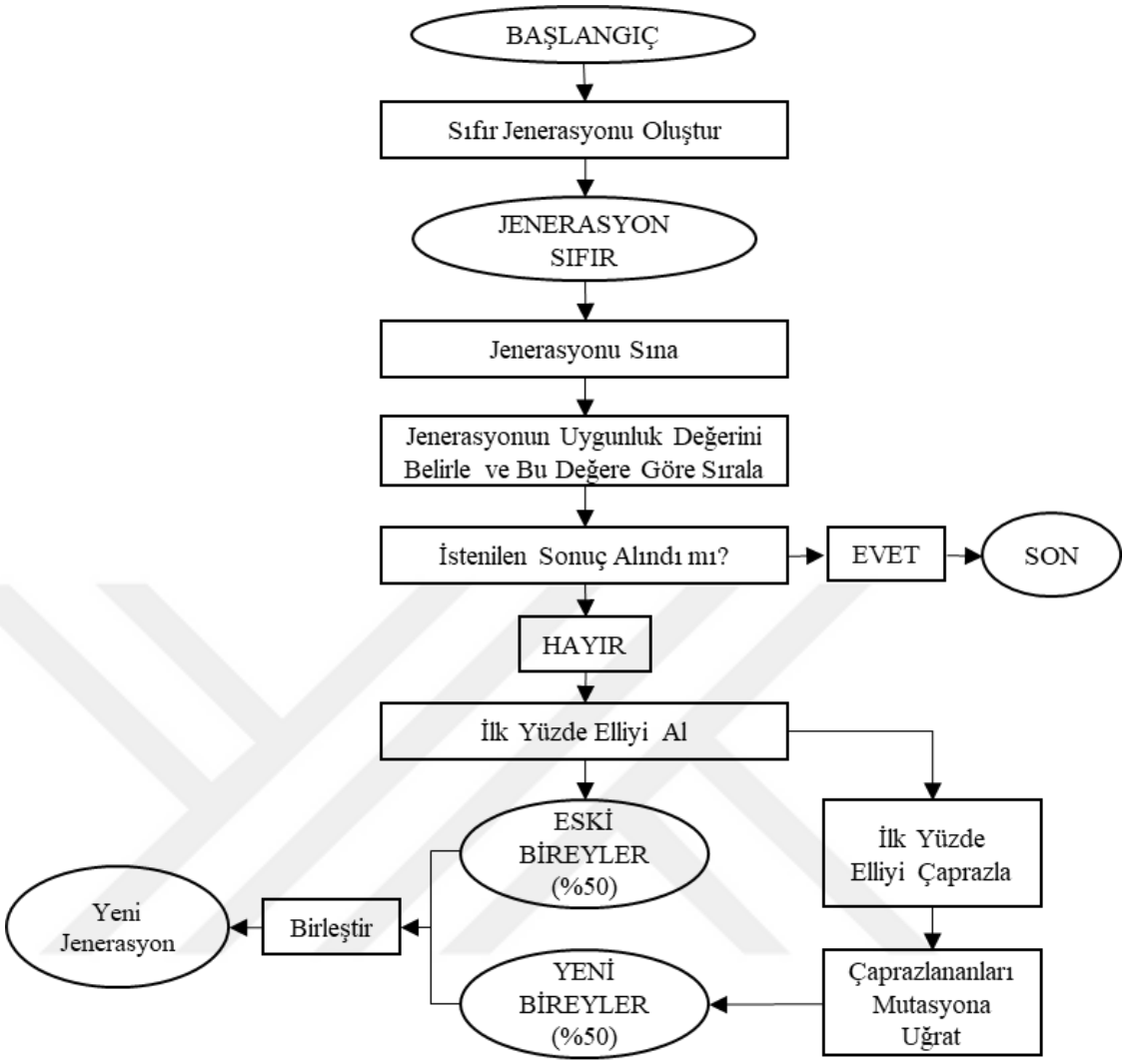
Mutasyon gerçekleştirilirken herhangi iki genin yeri değiştirilebilir, rastgele k tane genin değeri belirli bir oranda aşağı yukarı kaydırılabilir veya sıfırlanabilir. Probleme göre mutasyonun oranı ve uygulanma şekli değişiklik gösterecektir.

3.1.7 İterasyonlar

Evrimsel yapay sinir ağlarında bir jenerasyonun sınanıp, uygunluk değerlerinin belirlenip, çaprazlama ve mutasyon yapılıp, yeni bir jenerasyon oluşturulmasına kadar atılan tüm adımlar bir iterasyonu oluşturur (Şekil 3.8).

İterasyon döngüsünün adımları şunlarıdır:

1. Sıfır jenerasyonu oluştur.
2. Yeni jenerasyonu sına.
3. Sınanan jenerasyonu uygunluk değerlerine göre sırala.
4. Eğer istenilen değer alındıysa iterasyonları bitir yoksa adım 5'e devam et.
5. İlk yüzde elliyi sıradaki jenerasyona ekle.
6. İlk yüzde elliyi çaprazlayıp ve mutasyona uğratıp yeni yüzde elli elde et.
7. İki yüzde elliyi birleştirip yeni jenerasyon elde et ve adım 2'ye git.



Şekil 3.8 Evrimsel yapay sinir ağlarında iterasyon döngüsü

3.2 Evrimsel Yapay Sinir Ağı Kullanarak Oyun Yapay Zekası Geliştiren Benzer Çalışmalar

Evrimsel yapay sinir ağlar bu çalışmanın dışında başka çalışmalarda da bilgisayar oyunlarına yapay zekâ geliştirmek için kullanılmıştır. Yapay sinir ağları ile yapay zekâ önceden eğitilip oyunda kullanılmak üzere kaydedilmiş veya oyunda gerçek zamanlı olarak öğrenmesi sağlanmıştır.

3.2.1 Othello masa oyununa yapay zeka tasarlanması

Reversi olarak da bilinen Othello oyunu için yapılan bir çalışmada (Gunawan vd. 2012) yapay sinir ağları yardımı ile statik yöntemler kullanan Negamax adlı başka bir yapay zekâ sistemini yenen bir yapay zekâ geliştirilmiştir.

Othello oyunu iki kişi tarafından karşılıklı oynanan bir strateji masa oyunudur. Oyun 8x8 kare ızgara oyun alanında disk denilen siyah-beyaz taşlarla oynanır. Oyuncular toplamda 64 adet bir yüzü beyaz bir yüzü siyah olan diskleri kontrol eder. Bir taraf tüm diskleri beyaz, diğer taraf da siyah yapmaya çalışır. Bunun için oyuncu rakip oyuncunun disklerini yatay, dikey veya çapraz doğrultuda kendi iki diski arasına almalıdır. Oyunun amacı oyun sonunda en çok diske sahip olmaktır.

Gunawan vd. (2012) yirmi kişilik bir popülasyon ile eğitime başlamıştır. Popülasyon bireylerinin ağırlık değerleri Nguyen Widrow algortiması kullanılarak rastgele oluşturulmuş ve sonrasında popülasyon eğitilmeye başlanmıştır.

Nguyen Widrow Algoritamasının aşamaları şu şekildedir:

1. Nguyen Widrow faktörü tanımlanır.

$$\beta = 0{,}7\, p^{\frac{1}{n}}$$

$$n = Girdi\ katmanındaki\ düğüm\ sayısı$$

$$p = Gizli\ katmanlardaki\ toplam\ düğüm\ sayısı$$

2. Tüm ağırlıklara -0,2 ve 0,2 arasında rastgele değerler atanır.

3. Ağırlıklar için vektör uzunluğu belirlenir.

$$\|V_i\| = \sqrt{v_1^2 + v_2^2 +,,, + v_n^2}$$

$$v_i \ = i.ağırlık\ değeri$$

4. Yeni ağırlıklar güncellenir.

$$v_{i\ yeni} = \frac{\beta\ v_{i\ eski}}{\|V_i\|}$$

5. Eşik değeri hesaplanır.

$$Eşik = Rastgele(-\beta, \beta)$$

Popülasyon ağırlık değerleri hazırlandıktan sonra her birey turnuva tarzı bir sistemle birbiri ile yarıştırılır. Amaç turnuvadan şampiyon diye adlandırılan en iyi bireyi çıkarmaktır. Bir birey kazandığı maçtan 3, berabere kaldığı maçtan 1, kazanmadığı maçtan da 0 puan alır. Bireyler toplam puanına göre sıralanır ve şampiyon seçilir. Şampiyon sonrasında Negamax adlı yapay zeka ile 100 maç yapar. Eğer şampiyon 100 maçın 75'ini kazanabilir ise eğitim tamamlanmıştır. Eğer kazanamaz ise şampiyon bireyden yeni jenerasyon oluşturulur ve turnuva tekrar yapılır. Negamax maçların %75'inde yenilene kadar eğitimler ve turnuvalar devam eder.

Yapılan eğitimler sonucunda Gunawan vd. (2012)'nin yaptığı yapay zeka Negamax'i maçların %75'inde yenmeyi başarmıştır. Böylece arama ağaçlarında yapılan hesaplamalar azaltılmış, ağırlıklar yenilenerek ve şampiyon çaprazlanarak bir önceki nesilden daha iyi oynayan yeni nesiller elde edilmiştir.

3.2.2 NERO oyunu ile gerçek zamanlı yapay zeka eğitimi

Stanley vd. (2005)'nin yaptığı çalışmada rtNeat denilen bir evrimsel sinir ağı yöntemi kullanılarak oyuncunun gerçek zamanlı eğitebildiği, durumlara ayak uyduran ve inisiyatif geliştirebilen bir tür yapay zekâ geliştirilmiştir. Çalışmada NERO denilen ve bu amaç için geliştirilmiş olan bir oyun kullanılmıştır. NERO birincil kişi şahıs atış oyunudur ve oyuncu oyunu oynayarak robot askerleri savaşa hazırlamaya çalışmaktadır.

NERO oyunundaki her bir savaşçı robot, popülasyonun bir bireyi olarak işlev görmektedir. Savaşçı robotlar gerçek zamanlı olarak ve diğer robotlardan bağımsız sürede eğitilir. En kötü uygunluk değerine ait robot popülasyondan çekilir (eğer yeterli süre yaşamışsa) uygunluk değerleri tekrar belirlenerek uygunluk havuzu güncellenir. Havuzdaki en iyi bireyler çaprazlanarak yeni birey oluşturulur. Bu birey tekrar oyuna kazandırılır ve sonraki en kötü birey popülasyondan çekilene kadar eğitim devam eder.

En kötü bireyin seçilip elenebilmesi için o bireyin yeterince yaşamış olması gerekir. Diğer evrimsel yapay sinir ağı yöntemlerinde bireyler eşzamanlı olarak aynı süre eğitildiğinden eğitim iterasyonlar halinde devam ederken bu projede ise farklı zamanlarda bireyler doğar, yaşar ve ölür. Bu yüzden en kötü birey seçilirken yeterli süre yaşayıp yaşamadığına bakılmalıdır. Eğer birey yeni doğmuşsa daha puan toplamaya fırsatı olmamış demektir.

Kötü birey yeteri kadar yaşamışsa ve popülasyondan çekilmişse ortalama uygunluk değeri hesaplanan birey sayısı bir azalmış demektir. Bu yüzden ortalama uygunluk değeri tekrar hesaplanır ve böylece yeni doğacak birey için ebeveynlerin seçimi doğru bir şekilde yapılır.

Yeni bir birey oluşturulurken seçilecek ebeveyn rulet tekeri yöntemine göre seçilir. Yani bir bireyin ebeveyn olarak seçilmesinin olasılığının değeri, kendi ortalama uygunluk değerinin tüm ortalama uygunluk değerlerinin toplamına bölünmesi ile bulunur.

rtNEAT yönteminde ayrıca uygunluk eşiği denilen bir değer vardır. Bu değer oluşturulan popülasyonun sayısını dengede tutmak için kullanılır. Yeni birey üretildikten sonra uygunluk eşik değeri güncellenir. Eğer popülasyon çok kalabalık ise bu değer arttırılır ve eğer popülasyon nüfusu çok düşükse uygunluk eşiği değeri azaltılır. Böylece popülasyon nüfusu dengede tutulur.

Yeni birey eski bireyin yerine geçer ve böylece eğitimler devam eder. Ancak eğitimler gerçek zamanlı olduğu için değişimlerin arasında belirli bir süre geçmesi gerekir. Bu

süre hesaplanırken sabit bir zaman genişliği sayılarak hesaplanır. Bu zaman genişliğine tick denilmektedir. Örneğin bir saniyede bir tick geçtiğini varsayalım. Bir bireyin değiştirilmesi yüz tick ise yeni bireyler her yüz saniyede bir ortaya çıkacaktır. Bu yüzden belirli bir tick değerine ulaşılınca yeni bireyler oluşturulmalıdır. Eğer bireyler çok ender oluşturulursa evrim süreci yavaşlar. Tam aksine çok sık oluşturulursa da çaprazlamadan çıkacak yeni bireylerin bir önceki bireyden daha iyi olması olasılığı azalır. Bu yüzden tick iyi belirlenmelidir.

Neyse ki bu problemin çözümü için bir yöntem geliştirilmiştir. Popülasyon boyutu P, bir bireyin yaşaması gereken minimum tick m, değişimler arasındaki tick sayısı n ve son olarak da genç nüfusun (yaşama süresi az olan nüfusunun) oranı da I olarak alınmıştır. Bu durumda değişime uyumluluk formülü şu şekilde olur:

$$I = \frac{m}{P.n}$$

Yani popülasyon büyüdükçe ve değişim süresi arttıkça değişime açık bireylerin sayısı azalacaktır. Buna göre formül tekrar düzenlenerek n değeri gerçek zamanlı bulunabilir:

$$n = \frac{m}{P.I}$$

Böylece I ve m değeri belirlenerek ne kadar sıklıkla popülasyonun değiştirilmesi gerektiği bulunabilir. NERO oyununda bu değer %50 olarak belirlenmiştir.

NERO oyunu, oyuncuların iki aşamadan geçtiği bir oyundur. Birinci aşamada insan oyuncular savaşçı robotları eğitmek için eğitim alanı oluştururlar. Bunu gerçekleştirmek için insan oyuncular oyun alanına sabit veya hareketli olabilen statik yapay zekaya sahip taretler koyabilir, duvarlar yardımı ile labirentler inşa edebilir ve kaydırma çubukları yardımı ile oyun ayarlarını değiştirebilirler. Savaşçı robotlar oyuncunun kurduğu alanda gerçek zamanlı olarak eğitilir ve değerlendirilirler. Değerlendirme düşmana yaklaşma isabetli ateş etme, isabet alma, dostu takip etme ve kaçınma gibi birçok alanda yapılır. Hangi alanın ne kadar önemli olduğunu oyuncu kaydırma çubukları yardımı ile belirler ve ödüllendirme-cezalandırma sistemi tamamen oyuncunun elindedir.

Savaşçı robotlar iyi eğitilmeleri ve değerlendirilmeleri için oyun dünyasını iyi kavramalı ve oyun dünyasına doğru tepki vermeleri gerekmektedir. Bu yüzden savaşçı robotlar birçok sensöre ve harekete sahiptir. Düşman tespit radarları, imleç radarı, obje uzaklık sensörleri, atış hattı sensörleri savaşçı robotların sahip olduğu sensörlerdir. Bu sensörler savaşçı robotların sinir ağının girdi katmanını oluşturmaktadır. Savaşçı robotlar sağa-sola yürüme, ileri-geri yürüme ve ateş etme hareketlerine sahiptir. Bu hareketler savaşçı robotların sinir ağının çıktı katmanını oluşturacaktır.

Eğitimler tamamlandıktan sonra ikinci aşama olan savaş aşamasına sıra gelir. Bu aşamada oyuncu savaşçı robotların eğitimden nasıl yararlandıklarını görür. Savaş aşamasında eğitimi tamamlanan 20 robot iki takıma ayrılır ve takımlar birbiri ile ölümüne savaşırlar. Savaş bir taraf tamamen yok olana kadar veya kalan robotlar savaşamayacak durumda olana kadar devam eder. Savaş alanı ve takımların ayarlanması oyuncunun elinde olan bir şeydir.

Sonuç olarak çalışmaya katılan oyuncular NERO oyununu zevkli ve eğitici bulmuşlardır. Savaşçı robotlar gerçek zamanlı olarak eğitilmiş ve geliştiriciler tarafından ön görülemeyen yeni taktikler geliştirmişlerdir. Örneğin, kalabalık düşman grubuyla karşılaşan bir savaşçı robot aynı anda hem kaçmak hem de savaşmak için geri yürüme taktiğini bulmuştur. Geri yürüme taktiğiyle savaşçı robot düşmanlarla yüz yüze gelip ateş ederken geri geri yürüyerek düşmanlardan uzaklaşmayı başarmıştır.

4. MATERYAL VE YÖNTEM

YOGO'nun test edilmesi için bilgisayar oyunları dünyasında bir zamanlar popüler olan üç basit oyun ele alınmıştır. Flappy Bird, Google Dino Run ve Agar.io adlı üç oyuna YOGO yardımı ile üç çeşit yapay oyuncu üretilmesi hedeflenmiştir. Bu oyunların birer kopyası Unity Oyun Motoru içinde geliştirilip yine Unity Oyun Motorunda geliştirilen YOGO yardımıyla eğitilmiştir.

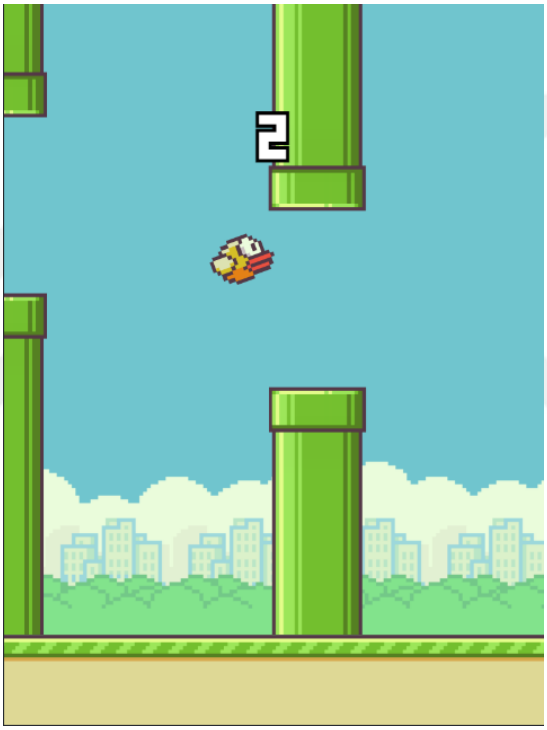
Bir oyun motoru kullanılıp oyunlar tekrardan geliştirilmesinin sebebi, herhangi bir oyunun kaynak kodlarına ulaşmanın birçok telif hakkı problemi ortaya koymasından ötürüdür. Phyton programlama dili ile görüntü işleme yöntemleri kullanarak herhangi bir ekrandaki oyundan girdileri çeken sistemler, yapay zekaya başka geliştiricilerin yaptığı oyunları oynamayı öğreten deneysel sistemlerdir. YOGO ise geliştiricinin kendi oyununa sorun yaşamadan uyarlayabileceği bir tür dinamik ortam olarak tasarlandığından oyunun direk kendi içine entegre edilmesi gerekmektedir. Bu yüzden Python gibi programlama dillerindeki görüntü işleme ve makine öğrenmesi kütüphaneleri yerine Unity kullanılmıştır.

Unreal Engine, Godot gibi diğer oyun motorları yerine Unity Oyun motorunun tercih edilmesinin sebebi ise Unity Oyun Motorunun tüm platformlara uyumlu olması ve iki boyutlu oyun tasarımı için kullanılan özelliklerin diğer oyun motorlarından daha geniş ve kullanışlı olmasıdır.

4.1 YOGO İçin Baz Alınan Oyunlar

Flappy Bird, Google Dino Run ve Agar.io oyunları YOGO'nun test edilmesi için baz alınan oyunlardır. Bu üç oyun da iki boyutlu olup geliştirilmesi kolay olduğu için seçilmiştir. Ayrıca YOGO'nun oyun ile bağlantı kurması için oyunda oyuncunun hareketleri ve dünya ile etkileştiği noktalar iyi tanımlanmalıdır. Bu üç oyunda da oyuncu hareketleri basit ve oyun dünyası ile etkileşim azdır. Bu yüzden YOGO'nun sınanması için uygun oyunlardır.

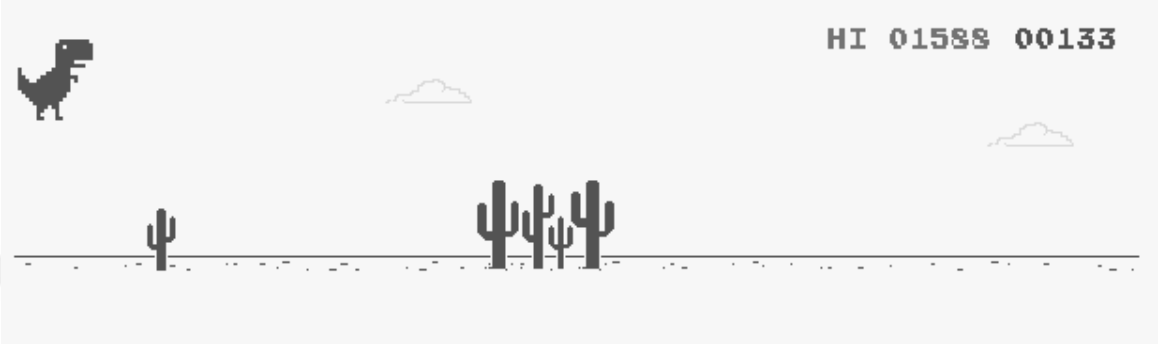
Flappy Bird oyunu, 2012 yılında Dong Nguyen tarafından geliştirilen yana kaydırmalı bir platform oyunudur (Warren C., 2014 ve bkz. Şekil 4.1.). Oyuncu tek bir tuşa basarak – veya ekrana dokunarak- oynattığı kuşun zıplamasını sağlar. Oyunda amaç yeşil boruların arasından geçerek havada kalmaya çalışmaktır. Oyuncu boruların arasından her geçişinde bir puan toplar ve oyuncu oynadıkça kuşun yatay hızı artar. YOGO için bu oyunu uygun yapan ögelerden biri oyuncunun sadece tek bir hamlesinin olması ve oyun dünyasından alınması gereken sadece beş değişkenin olmasıdır.



Şekil 4.1 Flappy Bird oyunundan bir ekran görüntüsü

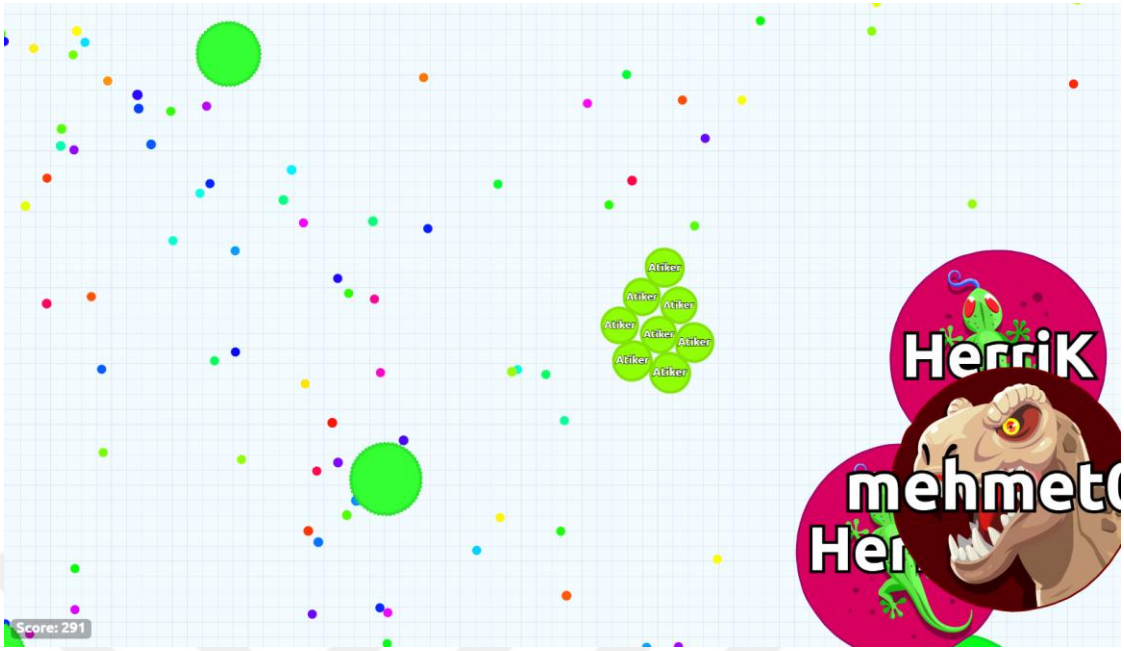
Google Dino Run oyunu, Google tarafından 2014 yılında Google Chrome İnternet Tarayıcısı için tasarlanmış bir oyundur (Wikipedia, 2022 ve bkz. Şekil 4.2.). Google Dino Run, internet bağlantısı olmadığı durumlarda Google Chrome İnternet Tarayıcısında çıkan ve boşluk tuşuna basılarak başlatılan bir oyundur. Flappy Bird gibi yana kaydırmalı platform oyundur. Oyuncunun amacı, çöl ortasında koşan bir dinozoru kaktüslerin üzerinden zıplatarak veya havada uçan dinozorların altından kaydırarak mümkün olduğunca fazla yol kat etmektir. Kat edilen yol ne kadar uzun ise oyuncunun aldığı puan o kadar fazladır. Google Dino Run, Flappy Bird oyununa kıyasla daha

karmaşık yapıdadır. Oyuncunun iki hareketi vardır: dinozoru zıplatmak veya kaydırmak. Ayrıca YOGO, yaklaşan kaktüslerin boyunu, genişliğini, uçan dinozorun yüksekliğini, dinozorun koşma hızını ve art arda gelen iki kaktüs kümesi arasındaki mesafeyi hesaba katmalıdır.



Şekil 4.2 Google Dino Run oyunundan bir ekran görüntüsü

Son olarak Agar.io oyunu oyuncuların birbirine karşı oynadığı bir tarayıcı oyunudur (Şekil 4.3.). Oyunda amaç kendinden küçük parçaları veya kendinden küçük oyuncuları yiyerek daha da büyümektir (Şafak I., 2015). Oyuncu ne kadar büyürse o kadar çok puan kazanır. YOGO için Agar.io benzeri Cell Rush isimli bir oyun tasarlanmış ve Agar.io tek oyunculu oynanabilir hale getirilmiştir. Oyuncunun hareketleri ve oyun dünyası ile etkileşim diğer iki oyuna göre çok daha fazladır ve Agar.io üç oyun arasından YOGO için en çok zorlayıcı olanıdır.

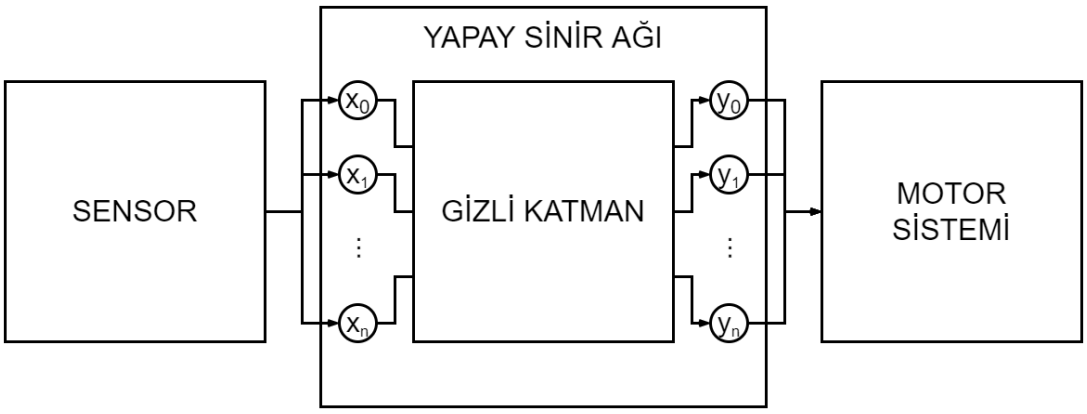


Şekil 4.3 Agar.io oyunundan bir ekran görüntüsü

Bahsi geçen üç oyunun benzer karşılıkları Unity Oyun motorunda geliştirilip YOGO sistemi ile birleştirilmiş ve üç oyun için de yapay oyuncu eğitilmeye çalışılmıştır.

4.2 Yapay Oyuncu

Yapay oyuncu, YOGO'nun en temel parçasıdır. Yapay oyuncu, verilen oyunu oynamak için eğitilmiş olan yapay zekadır. Yapay oyunculardan her birine yapay sinir ağı atanmıştır. Bu yapay sinir ağı oyun dünyasından aldığı verileri girdi olarak kullanır ve içerisinde işleyip çıktı üretir. Üretilen çıktı yapay oyuncunun hamlesini belirler. Yapay sinir ağından çıkan veriler yapay oyuncunun motor sistemine yollanır ve hareket hayata geçirilmiş olur. Oyun dünyasından girdi olarak veri alınmasını sağlayan yapay oyuncunun sensor sistemidir. Yapay oyuncunun temel çalışma prensibini gösteren şema Şekil 4.4'teki gibidir.



Şekil 4.4 Yapay oyuncu temel mimarisi

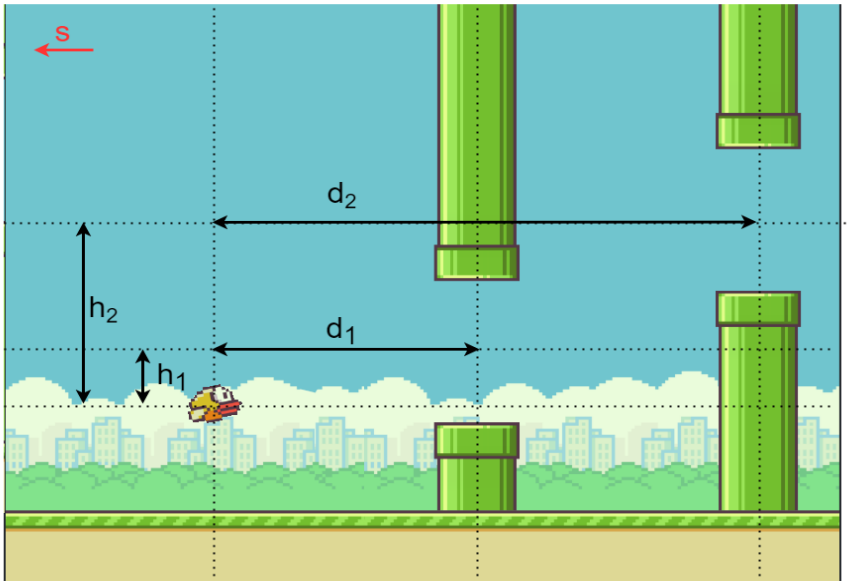
Yapay sinir ağı, yapay oyuncunun beyni görevini üstelenir. Burada yapay oyuncu, bir insan oyuncusunun oyunu oynarken yaptığı hareketlerden yola çıkarak tasarlanmıştır. Bir insan oyunu oynarken, ekrana bakıp görsel ve işitsel uyarıcıları işleyip bir tuşa basarak oyun dünyası ile etkileşime girer. Yapay oyuncu da sensor sistemi sayesinde oyun dünyasından aldığı verileri işleyip tepki oluşturmak üzere tasarlanmıştır.

Sensor, oyun dünyasına göre değişiklik gösterecektir. Burada YOGO mimarisini tasarlayacak geliştiricilerin sensor parçasının hangi verileri çekmesi gerektiğine karar vermesi gerekir. Motor sistemine gidecek olan verilerin, yapay sinir ağının çıktılarının, oyuncunun yapabileceği hareketleri temsil etmesi gerekir.

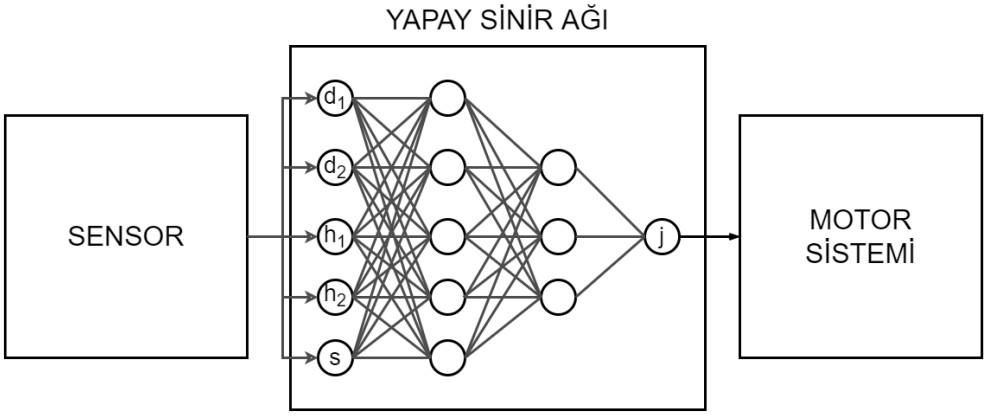
Flappy Bird oyununda sensor parçasının çekeceği beş adet değişken vardır (Şekil 4.5). Bu değişkenler kuşun en yakın boruya yatay uzaklığı (d_1), kuşun ikinci en yakın boruya yatay uzaklığı(d_2), kuşun en yakın borunun boşluğunun merkez noktası ile dikey uzaklık farkı (h_1), kuşun en yakın ikinci borunun boşluğunun merkez noktası ile dikey uzaklık farkı (h_2) ve kuşun yatay düzlemdeki hızı (s) olmaktadır. Böylece Flappy Bird oyununda, yapay oyuncunun sinir ağı beş adet girdi alacaktır.

Flappy Bird oyununda oyuncunun yapabileceği tek hamle kuşu zıplatmaktadır. Kuş zıplamadığı her an yerçekimi etkisi ile aşağı doğru düşecektir. Bu yüzden Flappy Bird oyunu için yapay oyuncunun motor parçasına gidecek tek veri zıplama (j) verisidir.

Flappy Bird oyunu için beş girdi alan, tek çıktısı olan iki gizli katmanı bulunan (ilk katman beş, ikinci katman üç düğümlü) bir yapay sinir ağı oluşturulmuştur. Yapay sinir ağının motor ve sensor parçalarını barındıran şeması Şekil 4.6'da gösterildiği gibi olacaktır. Burada Yapay sinir ağırlığının düğümleri arasındaki toplam bağlantı 45 adet (5 + 5x5 + 5x3 = 45) olacaktır. Bu 45 adet bağlantı yapay sinir ağının ağırlıklarından oluşan bir dizidir. Genetik algoritma yöntemi ile değerleri optimize edilecek olan bu ağırlıklar dizisi yapay oyuncunun davranışlarını belirleyecektir.



Şekil 4.5 Flappy Bird yapay oyuncunun sensor parçasının oyun dünyasından aldığı verileri gösteren şekil



Şekil 4.6 Flappy Bird yapay oyuncusu diyagram şeması

Google Dino Run oyunu için geliştirilen yapay oyuncu ise daha fazla değişkeni girdi olarak alacaktır (Şekil 4.6.). Toplam 10 adet girdi vardır ve oyuncu iki hamle yapabildiği halde bu iki hamleden sadece birini yapabilmektedir ve bu hamleler birbirinin zıttıdır. Bu yüzden sadece tek bir çıktı vardır. Yapay sinir ağın girdileri ve çıktıları aşağıdaki gösterildiği gibidir.

Girdiler:

1. Dinozor ile en yakın kaktüs kümesi arasındaki yatay uzaklık. (d_1)
2. Dinozor ile ikinci en yakın kaktüs kümesi arasındaki yatay uzaklık. (d_2)
3. Dinozor ile Pterodactyl arasındaki yatay uzaklık. (d_3) (Eğer oyunda dinozorun sağında herhangi bir Pterodactyl yoksa uzunluk 10000 olarak alınır.)
4. Dinozora en yakın kaktüs kümesinin dikey uzunluğu (h_1)
5. Dinozora en yakın ikinci kaktüs kümesinin dikey uzunluğu (h_2)
6. Pterodactyl ile dinozor arasındaki dikey uzaklık (h_3)
7. Dinozora en yakın kaktüs kümesinin yatay uzunluğu (l_1)
8. Dinozora en yakın ikinci kaktüs kümesinin yatay uzunluğu (l_2)
9. Dinozorun koşma hızı (s)
10. Dinozorun yerden yüksekliği (H)

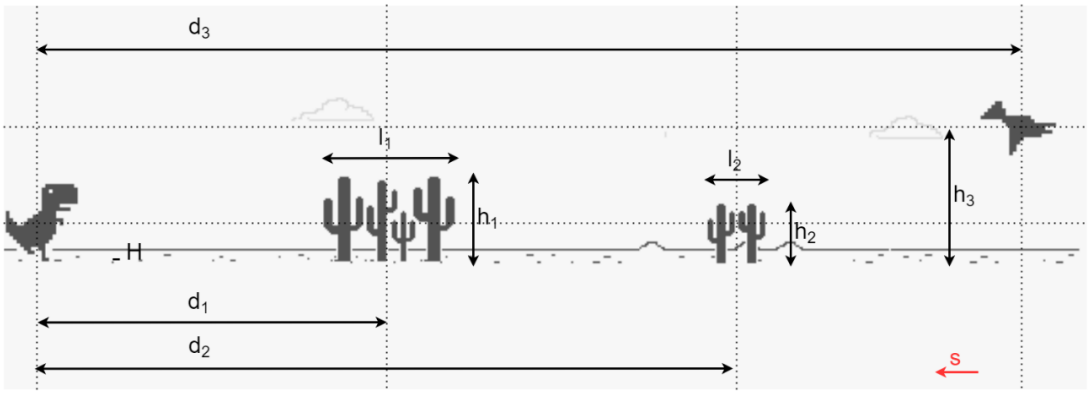
Çıktı:

1. Dinozorun zıplaması veya eğilmesi. (j) Bu değişkenin değerine göre motor aşağıdaki hareketleri yapar:

$$j < -0.33 \quad ise \quad Eğil$$

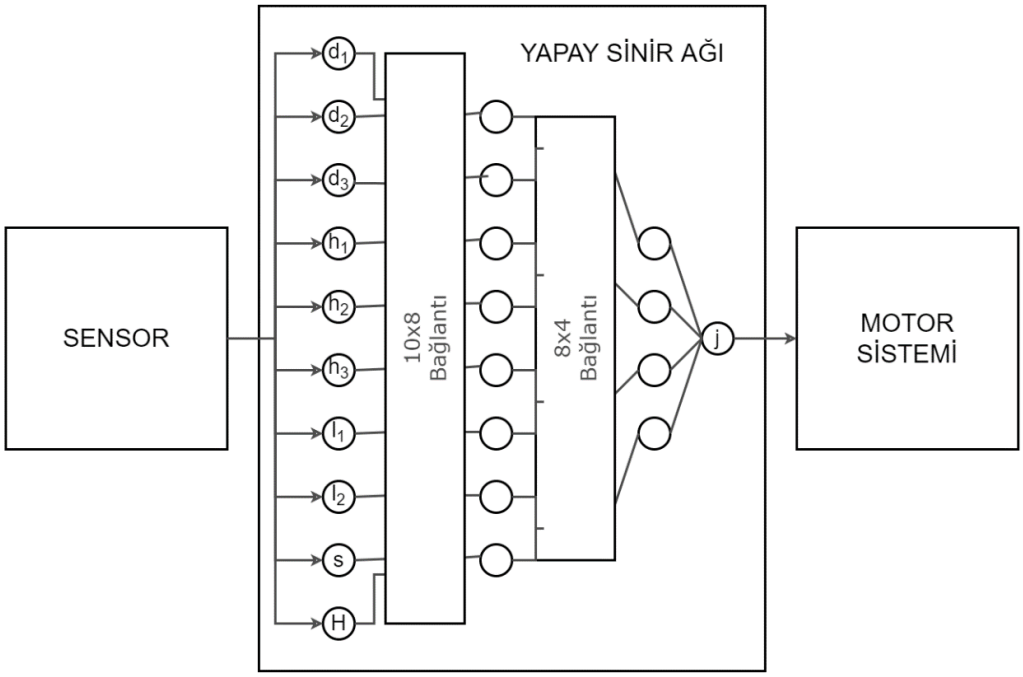
$$-0.33 \leq j \leq 0.33 \quad ise \quad Bir\ Şey\ Yapma$$

$$j > 0.33 \quad ise \quad Zıpla$$



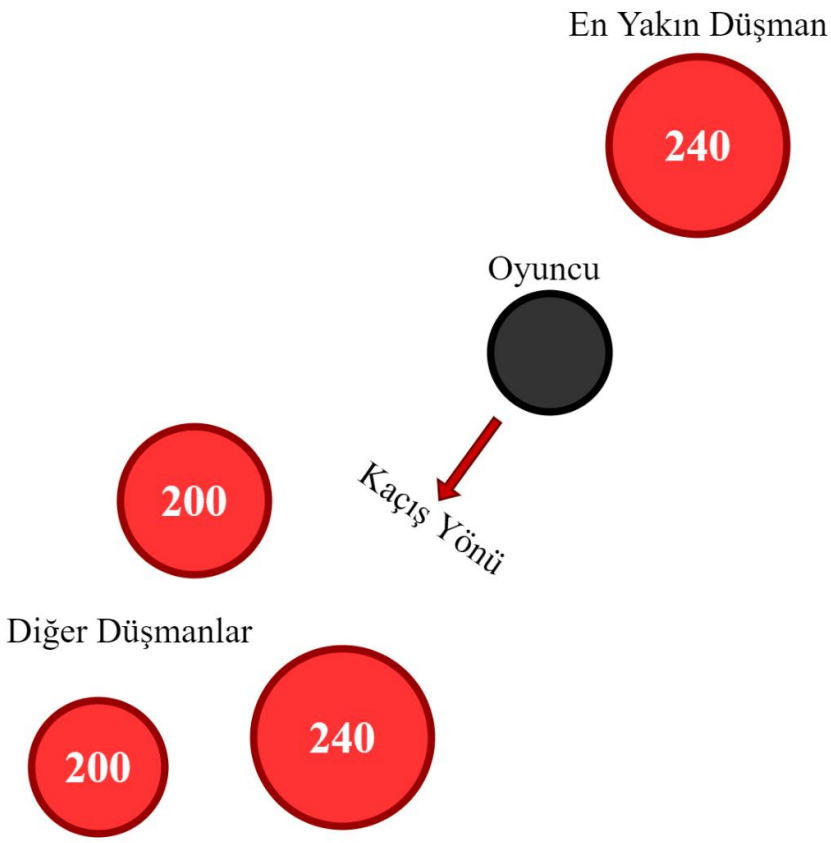
Şekil 4.7 Google Dino Run yapay sinir ağı girdilerini gösteren şekil

Google Dino Run yapay oyuncusunun yapay sinir ağında 10 girdi, 1 çıktı ve 2 gizli katman (Birinci katmanda sekiz, ikinci katmanda dört adet düğüm bulunur) bulunmaktadır. Toplam ağırlık sayısı 116 adet (10 x 8 + 8 x 4 + 4 x 1 = 116) olacaktır (Şekil 4.7).



Şekil 4.8 Google Dino Run yapay oyuncusu diyagram şeması

Agar.io benzeri oluşturulan Cell Rush adlı oyunun yapay oyuncusu önce bahsedilen diğer iki yapay oyuncudan çok daha karmaşık yapıdadır. Öncelikle sensor parçasının oyun dünyasından aldığı verilerin hesaplanması diğerlerine göre daha karmaşıktır. Sadece oyuncunun en yakın olduğu hedefleri (düşman veya yiyecek) yapay oyuncunun içinde bulunduğu durumu kavramasını engelleyecektir. Örneğin Şekil 4.8'deki gibi bir durum söz konusu olsaydı ve eğer yapay oyuncu sadece en yakındaki tehdidi hesaba katıp o tehdidin zıttı bir yönde kaçsaydı diğer tehditlerin arasına doğru kaçmak zorunda kalır ve daha zor bir duruma düşerdi. Ancak yapay oyuncunun tüm tehditleri değerlendirip en optimal yönde ilerlemesi gerekir ki zor durumlardan kaçabilsin. Aynı senaryo avını kovalarken de ortaya çıkar. En yakındaki cılız veya yakalaması zor hedefi kovalamasından ise biraz daha uzaktaki ama daha çok besin getirecek hedefi kovalaması gerekmektedir.

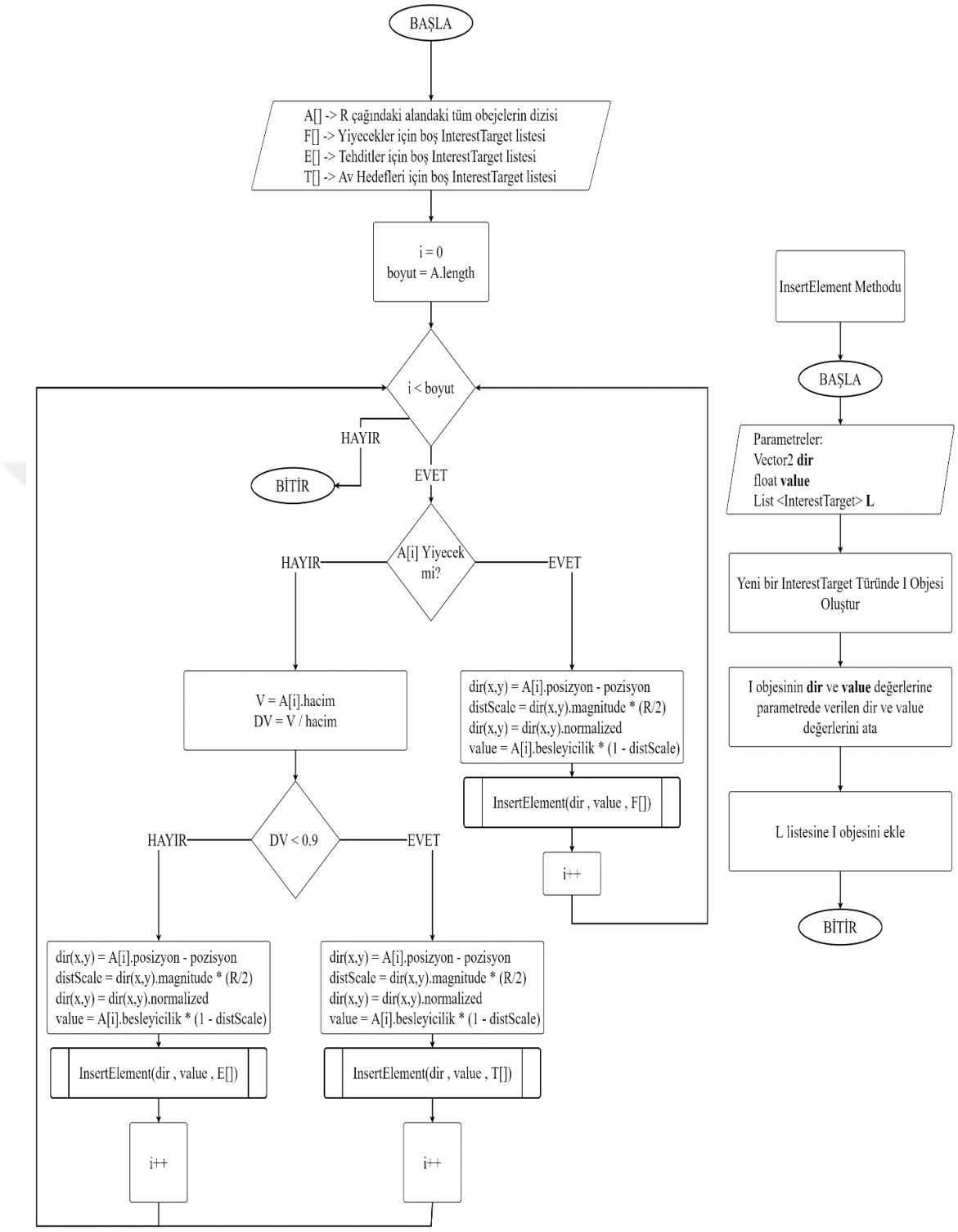


Şekil 4.9 Cell Rush oyununda oyuncunun sadece en yakın düşmanı hesaba katması sonucu oluşacak sorunu gösteren şema

Bu yüzden yapay oyuncu sensor parçası oyuncunun belirli bir yarıçapındaki tüm objeleri gruplara ayırıp bu gruplar arasında hesaplamalar yaparak yön belirlemelidir. Bu sebeple yapay oyuncu üç adet liste ile işlem yapmaktadır. Bu listelerden ilki alan içerisindeki hareketsiz yiyeceklerin listesi olan F listesi. İkincisi, oyuncudan daha zayıf olan diğer oyuncuların listesi olan T listesi. Son olarak da oyuncuya tehdit olan diğer oyuncuların listesi olan E listesidir. Yapay oyuncu bu listeleri oluşturmak için öncelikle merkezi kendi pozisyonu olan R çapındaki dairesel alandaki tüm objeleri A dizisi içerisine aktarır ve hepsinin türü "GameObject" olan bu elemanlar ayıklama algoritması ile önce bahsedilen üç listeye ayrıştırılacaktır. Tüm Gameobject elemanları ayıklama algoritmasından geçerek ilgili listelere "InterestTarget" objesi olarak kaydedilir. InterestTarget objesi iki değişken barındırır. Birinci değişken iki boyutlu bir vektör verisi olan "dir" değişkenidir. İkinci değişken ise bu yönün ağırlığı olan "value" değişkenidir. Value değişkeni yemek hedefleri için pozitif, tehdit hedefleri için negatif olacaktır.

A dizisinin elemanlarını ayrıştırmak için kullanılan ayırma algoritması Şekil 4.9'daki adımları izler. İlk önce A dizisindeki objenin türüne bakılır. Eğer obje yiyecek objesi ise yön hesaplanır ve yeni oluşturulan InterestTarget objesinin dir değişkenine atanır. Sonra yiyecek objesinin besleyicilik değeri ile 1 - distScale değişkeni çarpımı aynı InterestTarget objesinin value değişkenine atanır. Büyük planktonlar için besleyicilik değeri 10, küçük planktonlar için 5, büyük sitoplazma parçaları için 50 ve küçük sitoplazma parçaları için 20 olacaktır.

Eğer ayrıştırılacak obje başka bir oyuncu ise rakip oyuncunun hacmi ile oyuncunun kendi hacmi kıyaslanır. Eğer rakibin hacmi oyuncunun hacminin yüzde doksanından daha büyükse bu rakip tehdit olacak aksi takdirde rakip hedef olacaktır. Rakibin besleyicilik değeri rakibin hacminin 10'da biridir. Yine rakibin yönü yeni oluşturulan InterestTarget objesinin yönüne atanır ve rakibin besleyicilik değeri ile 1 - distScale değişkeni çarpımı InterestTarget objesinin değer değişkenine atanacaktır. Eğer rakip hedef ise sayı olduğu gibi aktarılacak, değilse besleyicilik değerinin negatifi değer değişkenine atanacaktır.



Şekil 4.10 A dizisini gruplara ayırmak için kullanılan ayırma algoritmasının akış diyagramı

A dizisindeki objelerin Yön vektörü hesaplanırken aşağıdaki formüle kullanılır:

$$dir(x,y) = other(x,y) - player\ (x,y)$$

$$distScale = dir(x,y).magnitude * \frac{R}{2}$$

$$dir(x,y) = Normalize(dir\ (x,y))$$

Hedef objenin pozisyon other(x,y) vektörü ile, oyuncunun kendi pozisyon player (x,y) vektörü arasındaki fark hedefe doğru bir ağırlıklı yön vektörü verecektir. Bu vektörün büyüklük değerinin taranan alanın yarıçapına bölünmesi sonucu elde edilen değişken InterestTarget objelerinin value değişkeninin oluşturulmasında kullanılacaktır. Normalize() fonksiyonu ile bu vektör birim vektöre dönüştürülerek sadece yön belirten dir(x,y) vektörünü oluşturacaktır.

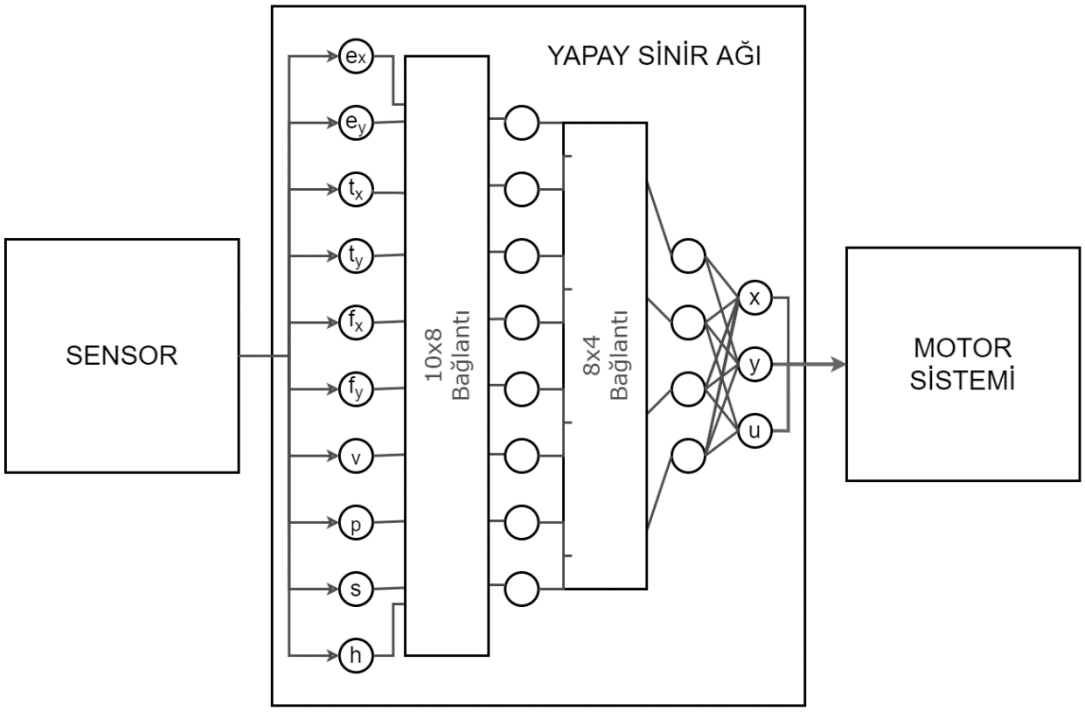
A dizisi gruplara ayrıldıktan sonra F, E ve T listesinde yüzlerce eleman olabilir. Tüm bu elemanların pozisyon vektörü (2 değişken) ve değer değişkeni yapay sinir ağına yollanacak olsaydı yüzlerce girdi olurdu ve her seferinde girdilerin sayısı değişirdi. Yapay siniri ağı için bu durum ideal değildir. Onun yerine listelerin her biri vektöre etme metodu ile birer vektöre indirilir ve eleman sayısı ne olursa olsun sadece altı adet değişken ile temsil edilecektir. Vektöre etme metodu liste içerisindeki InterestTarget objelerinin her birinin yön vektörünü (dir(x,y)) değer değişkeni (value) ile çarparak başta (0,0) olan toplam vektörüne ekler. Böylece tüm elemanların ortalama ağırlığı bulunmuş olur ve o yöne doğru ağırlıklı bir vektör elde edilir.

Vektöre Etme Metodunun adımları:

1. İlk değeri (0,0) olan iki boyutlu bir toplam vektörü oluştur.
2. İlk değeri sıfır olan bir index değişkeni oluştur.
3. Index değeri listenin eleman sayısından az olduğu müddetçe adım 4, 5 ve 6'yı tekrarla
4. Liste elemanının dir (x,y) vektörü ile value değişkeninin skalar çarp.
5. Oluşan yeni vektörü toplam vektörüne ekle
6. Index değişkenini bir arttır.
7. Toplam Vektörünün x ve y değişkenleri yeni girdi değerleridir.

E listesinden oluşturulan toplam vektörünün değerleri (e_x, e_y), F listesinden çıkan toplam vektörünün değerleri (f_x, f_y) ve T listesinden çıkan toplam vektörünün değerleri (t_x, t_y) yapay sinir ağın altı girdisi olacaktır. Bu girdilere binaen oyuncunun hacmini veren v değişkeni, hızını veren s değişkeni, süper gücünün hazır olup olmadığını ve hazır değilse ne kadar süresinin kaldığını (yüzde cinsinden) gösteren p değişkeni ve oyuncunun açlık seviyesini gösteren (0 = hiç aç değil, 1= son derece aç) h değişkeni de girdiler arasındadır. Böylece yapay sinir ağının toplam on adet girdisi vardır.

Yapay sinir ağının çıktı olarak üç değişkeni bulunmaktadır. Bunlardan ikisi oyuncunun hareket etmesi gereken yönü ifade eden x ve y değişkenleridir. Sonuncusu da güç kullanıp kullanmadığını belirten u değişkenidir. (Şekil 4.10)



Şekil 4.11 Cell Rush yapay oyuncusu diyagram şeması

4.3 Yapay Oyuncu Geliştirme Ortamı (YOGO)

YOGO olarak adlandırılan sistem, evrimsel yapay sinir ağları tekniklerini kullanan oyunlara uygulanabilir, dinamik yapıya sahip bir sistemdir. Problemin çözümü için geliştirilmiş olan YOGO içeriğinde yapay oyuncuları barındıran ve bu oyuncuları evrimsel yapay sinir ağı yöntemiyle eğiten sistemin genel adıdır.

YOGO ön kurulumunu yaparken belirlenmesi gereken değişkenler ve yapılar vardır. Bunlardan en önemlisi yapay oyuncunun sinir ağıdır. Yapay sinir ağı, tüm yapay oyuncular için belirlenmesi gereken bir yapıdır. Baz alınan oyunların hepsi tek bir tip yapay oyuncu sinir ağı gerektirir. Bu çalışmada kullanılacak yapay oyuncuların modeli bölüm 4.2'de ayrıntılı olarak belirtilmiştir.

YOGO ön kurulumunda popülasyon sayısına, mutasyon oranına, sınır jenerasyona, çaprazlama tipine ve seçilim tipine karar verilmelidir. Çalışmada baz alınan tüm oyunlarda tekdüze (uniform) çaprazlama ve sıralı seçim tekniği kullanılmıştır. Flappy Bird oyunu için 32 ve 64 bireylik, Dino Run oyunu için 32 bireylik, Cell Rush oyunu için de 8,16 ve 32 bireylik popülasyonlar sınanmıştır. Flappy Bird ve Dino Run oyununda her jenerasyon en fazla iki dakika sınanmıştır. (Eğer jenerasyonun tüm bireyleri oyun içinde yanarsa iki dakikanın dolmasına gerek yoktur.) Cell Rush oyununda ise oyun kuralları gereği 15 dk sınır konulmuştur.

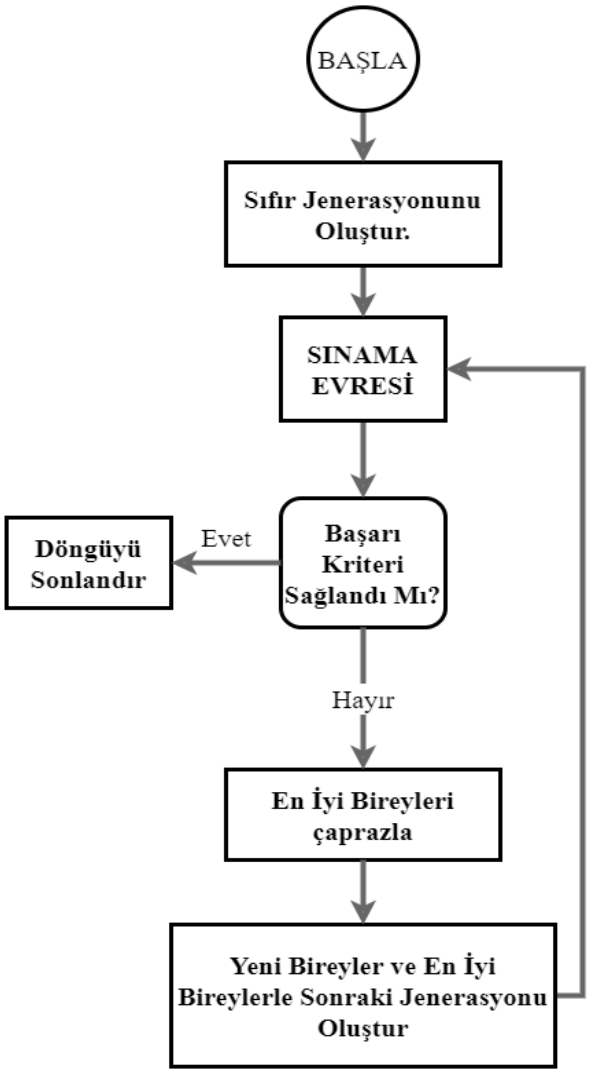
Her bir oyun için uygunluk fonksiyonu ve uygunluk değeri hedefi farklıdır. Ayrıca yapay oyuncuların ulaşması gereken bir başarı kriteri vardır. Bu kriter uygunluk değeri belirli bir puana ulaşınca veya oyuncu sınama süresinin sonuna kadar hayatta kalınca sağlanmış olur. Başarı kriteri Flappy Bird ve Dino Run için süre dolana kadar hayatta kalmak olurken, Cell Rush oyunu için on beş dakikada belirli bir skora ulaşmak olmuştur. Bunun nedeni Cell Rush oyununda yanma durumunun Falppy Bird ve Dino Run'daki gibi olmamasındandır.

4.3.1 YOGO ana döngüsü

YOGO ana döngüsü, evrimsel sinir ağlarındaki döngü ile aynıdır. İlk önce sıfır jenerasyonu oluşturulur. Bu jenerasyon sınanır ve değerlendirilir. Eğer jenerasyon başarı kriterini sağlamıyorsa bireyler çaprazlanarak yeni jenerasyon oluşturulur ve sınama tekrarlanır. Başarı kriteri sağlanana veya sınır iterasyona ulaşılana kadar döngü devam eder.

Döngünün sınama kısmı, yapay oyuncular süre sınırı aşana kadar veya tüm oyuncular yanana (eğer oyunda yanma durumu varsa) kadar devam eder. Flappy Bird oyunu ve Dino Run oyunu için yapılan testlerde sınama kısmı maksimum iki dakika olarak ayarlanmıştır. Bu iki dakikalık süre ayrıca başarı kriterinin de belirlenmesine yol açmıştır. Cell Rush oyununda ise oyuncular on beş dakika boyunca mümkün oldukça fazla skor almaya çalışırlar. Oyun esnasında ölen oyuncular uygun bir yerde tekrar doğduklarından on beş dakika boyunca (yeniden doğma süresi hariç) oyuna dahil olurlar. Bu yüzden Cell Rush oyununda hayatta kalmak bir başarı kriteri olmamıştır.

Sınama sonunda başarı kriterine ulaşan oyuncular varsa karara göre halihazırdaki oyuncuları geliştirmek için döngüye devam edilebilir veya döngü orada sonlanarak başarı kriterine ulaşan bireyler incelenebilir. Bu karar geliştiricinin kendisine kalmıştır. Sınama sonunda hiçbir birey başarı kriterini sağlamıyorsa bireyler çaprazlanır ve yeni bir jenerasyon oluşturulur. Bunun için en iyi yüzde elli birey çaprazlamaya girer ve sonra sıradaki jenerasyona dahil edilirler. Çaprazlamadan doğan yeni bireyler de yeni jenerasyonun üyesi olurlar. En iyi yüzde elliye girmeyen bireyler yok edilir ve bir sonraki jenerasyona dahil olmazlar. YOGO sistemindeki ana döngü Şekil 4.11'de gösterildiği gibidir.



Şekil 4.12 YOGO ana döngüsü

4.3.2 YOGO'nun tasarlanması

YOGO mimarisi tasarlanırken aşağıdaki sorulara cevap verilmelidir:

1. Yapay oyuncunun hareketleri nelerdir?

Oyuncu, oyundaki amacına ulaşmak için ne yapmalıdır. Yani eğitilecek yapay oyuncunun gerçek bir oyuncuya benzemesi için nasıl hareketlerde bulunması ve ne biçimde oyunu oynaması gerekir?

2. Yapay oyuncu oyun dünyasını nasıl görür?

İnsan oyuncular ekranda gördüğü nesnelere ve efektlere göre tepki gösterir ve oyunu oynarlar. Yapay oyuncunun oyun dünyasında etkileşmesi gereken etkenler nelerdir ki gerçek bir oyuncu gibi dünyaya tepki verebilsin?

3. Yapay oyuncunun beyni nasıl olmadır?

Yapay oyuncunun sinir ağı birinci ve ikinci soruya değinilerek nasıl yapılmalıdır? Girdiler oyun dünyasıyla etkileşimine göre olacak, çıktılar yapay oyuncunun alabileceği hareketleri temsil edecektir.

4. Yapay Oyuncu için başarı kriterleri nelerdir?

Yapay Oyuncunun bir insan gibi oynayıp oynamadığına nasıl karar verilebilir? Yani yapay oyuncu hangi seviyeye ulaştığında yapay oyuncunun oyunu oynama şekli tatmin edici bir seviyede olacaktır?

5. Yapay oyuncunun başarı kriterine ne kadar yakınlaştığı nasıl ölçülür?

Yapay oyuncu için nasıl bir uygunluk fonksiyonu oluşturmalıdır ki performansı ölçülüp başarı kriterine yaklaşıp yaklaşmadığı anlaşılsın?

6. Her bir sınama ne kadar sürecektir?

Yapay oyuncu değerlendirmesi bitmeden önce ne kadar süre oyunda kalacaktır? Değerlendirmesi biten oyuncunun uygunluk değeri incelenip başarı kriterine ulaşıp ulaşılmadığına bakılacaktır. Ayrıca uygunluk değerine göre bir sonraki nesle geçip geçmeyeceği de belirlenecektir.

7. En fazla kaç jenerasyon oyuncular sınamadan geçilmelidir?

Yapay oyuncunun, başarı kriterine ulaşması en fazla kaç jenerasyon sürmelidir?

8. Yapay oyuncunun kalıtımı (genotip) sonraki nesillere nasıl geçmelidir?

Yapay oyuncuların çaprazlanması, çaprazlanma sonucu ortaya çıkan bireylerin mutasyona uğraması ve yeni jenerasyonun oluşturulmasında hangi teknikler kullanılmalıdır?

9. Popülasyon büyüklüğü ne kadar olmalıdır?

Her bir jenerasyonda ne kadar yapay oyuncu sınanmalıdır. Eğer popülasyon sayısı çok büyük tutulursa kare başı yapılan hesaplamalar artacaktır. Popülasyon sayısı çok az tutulursa çözüme ulaşılma süresi uzayacaktır.

4.3.3 Flappy Bird oyunu için YOGO tasarlanması

Flappy Bird oyununda oyuncunun tek bir hareketi bulunmaktadır: Zıplamak. Bu yüzden yapay oyuncunun sinir ağında tek bir çıktı olur. Bölüm 4.1'de bahsedildiği üzere yapay oyuncunun sinir ağının beş girdisi bulunmaktadır.

Uygunluk değeri her bir yapay oyuncu için geçilen boru sayısı yapılabilir. Ancak bu durumda çoğu oyuncunun uygunluk değeri aynı veya çok yakın olacaktır. (Özellikle baştaki jenerasyonlarda) Bu durumda da en iyi bireylerin seçilmesini zorlaştıracaktır. O yüzden alternatif olarak hayatta kalınan süresi (Saniye bazında) uygunluk değeri için daha doğru bir tercih olabilir. Ancak yapılan testlerin sonunda iki fonksiyondan da farklı uygunluk fonksiyonu tercih edilmiştir:

$$F = 4T + 10P$$

Bu formülde F = uygunluk puanı, T = Oyuncunun en yakın borunun merkez noktasıyla aynı hizada durma süresi, P = Başarılı şekilde geçilen boru sayısıdır. Oyuncu bir sonraki borunun merkezinin hizasında kaldığı müddetçe puan biriktirmektedir. Aynı zamanda geçilen her bir boru oyuncuya puan kazandıracaktır. Bu fonksiyonu kullanmadaki amaç oyuncuların performansını daha isabetli bir şekilde hesaplamaktır.

Bir sınama döngüsü bir dakika ile sınırlıdır. Oyuncular bir dakika oyunu oynadıklarında yaklaşık 18 adet boru geçeceklerdir. Eğer bu sürenin %50'sinde sıradaki boruların merkeziyle aynı hizada kalabilirlerse oyuncular bir dakikada toplam 300 uygunluk puanı toplayacaklardır. Bu yüzden hedef uygunluk puanı 300 belirlenmiştir.

Flappy Bird oyunu için 32 nüfuslu bir popülasyonla eğitim yapılmıştır. 32 nüfuslu popülasyon 188. jenerasyonda başarı kriteri sağlamıştır. Flappy Bird oyunu için maksimum iterasyon sayısı 1000 olarak belirlense de başarı kriterine 188. Jenerasyonda ulaşıldığı için simülasyon 1000. Jenerasyona ulaşmadan sonlandırılmıştır.

Flappy Bird oyununun yapay oyuncuları için genotip 48 genden oluşmaktadır. Bu genotip için gen sayısı ve yeri önemli olup sırası önem arz etmemektedir. Her bir gen özgün olmayan ve -1 ile 1 arasında değişen float değere sahiptir. Bireyler için çaprazlama şekli tekdüze çaprazlamadır ve iki ebeveynden iki adet birey ortaya çıkar. Seçilim tipi sıralı seçilim olduğundan çözüme ulaşma süresi ile çeşitlilik arasında bir denge oluşturulmak istenmiştir. Ancak problemin basitliğinden ötürü son jenerasyondaki bireylerin hemen hemen hepsi aynı oynayış şekline ve performansa sahiptir. Mutasyon oranı sıfır jenerasyonda yüzde 1 olarak belirlenmiştir. Eğer ardışık jenerasyonların performansları arasında fark olmazsa mutasyon şansı artacak ve performans iyileşmeye başlayınca tekrar varsayılan değerine (%1'e) düşürülecektir.

4.3.4 Dino Run oyunu için YOGO tasarlanması

Google Dino Run oyununda, oyuncu ya zıplayabilir ya da eğilebilir. Dinozor ekranın solundan sağına otomatik koşmaktadır ve oyun ilerledikçe koşu hızı artmaktadır. Yapay oyuncunun hareketi tek bir değişkende birleştirilmiştir: Zıplama. Eğer zıplama değişkeninin değeri 0.33'ten büyük ise yapay oyuncu zıplar. Eğer değişken -0.33'ten küçük ise yapay oyuncu eğilir. Eğer değer -0.33 ten 0.33'e kadar bir değer ise yapay oyuncu bir harekette bulunmaz.

Yapay oyuncu oyun dünyasını algılayabilmek için önündeki ilk iki kaktüs kümesinin yükseklik ve genişlik değeriyle bu kaktüs kümelerinin ne kadar uzakta olduğunu gösteren uzaklık değerini kullanır. Bu değerlere binaen karşıdan üç farklı yükseklikte gelme ihtimali olan Pterodactyl türünde uçan bir dinozorun yüksekliğini ve oyuncu ile uçan dinozor arasındaki mesafeyi veren değerleri alacaktır. Ancak oyunun çoğu anında Pterodactyl olmayacaktır. Pterodactyl olmadığı anlarda yükseklik 0, uzaklık da 10000 olarak alınır. Yani Pterodactyl çok uzaktaymış gibi davranılır. Oyuncunun yerden yüksekliği ve koşma hızı da oyun dünyasını algılamak için alınan değerlerdir.

Girdiler ve çıktı göz önüne alındığında Google Dino Run oyunu için yapay oyuncunun sinir ağı 10x8x4x1 şeklinde bir yapay sinir ağı olacaktır. Yapay oyuncu toplam 126 adet ağırlık değeri olan yapay sinir ağına sahiptir.

Google Dino Run oyununda skor kat edilen mesafeye göre verilmektedir. Oyuncu kat ettiği her birim mesafe için puan alır. Yapay oyuncu için de bu puan uygunluk değerini oluşturmada yardımcı olacaktır. Skora binaen uygunluk puanı hesaplanırken geçilen engeller de fonksiyona katılacaktır. Buna göre uygunluk puanı:

$$F = D + 10\ x\ C + 20\ x\ P$$

Olacaktır. Burada D = Kat edilen mesafe, C= Geçilen kaktüs Sayısı (Kaktüs grupları bir sayılır, uzun kaktüsler 2 kat puan kazandırır.) ve P = Geçilen Pterodactyl sayısıdır.

Başarı kriteri 90 saniye hayatta kalmak olduğu için yapay oyuncular sadece 90 saniyelik bir sınamadan geçirilirler. Sonrasında uygunluk değerleri analiz edilir ve sonuca göre döngü bitirilir veya sonraki jenerasyon üretilir. 90 saniyede bir oyuncunun alabileceği minimum puan (hesaplanan yaklaşık puan) 1200 puan hesaplanmıştır. Uygunluk hedefi de bu yüzden 1200 puan olarak belirlenmiştir.

Google Dino Run oyunu, Flappy Bird oyunundan genel olarak daha zor ve karmaşık bir oyun olduğu için hedef kriterine ilk denemede 316. Jenerasyonda, ikinci denemede ise

217. Jenerasyonda varılmıştır. İlk denemede popülasyon 32 iken, ikinci denemede 64 alınmıştır.

Dino Run oyunu için eğitilen yapay oyuncuların Flappy Bird oyununa geliştirilen yapay oyunculardakine benzer bir genotip yapısı vardır. Flappy Bird yapay oyuncularından farklı olarak Google Dino Run oyunu için geliştirilen yapay oyuncular 126 genli genotipe sahiptir. Dino Run yapay oyuncuları tekdüze (uniform) çaprazlama tekniği ile çaprazlanacak, rulet tekeri yöntemi ile seçileceklerdir. Mutasyon oranı Flappy Bird Oyununda olduğu gibi %1 başlayacak ve performansa göre arttırılıp azaltılacaktır.

4.3.5 Cell Rush oyunu için YOGO tasarlanması

Cell Rush oyununda Agar.io oyunundaki gibi oyuncu 2 boyutlu düzlemde hareket edebilir. Ancak oyuncu Cell Rush oyununda Agar.io oyunundaki gibi bölünemez veya parça atamaz. Bunun yerine özel güç kullanabilir. Özel güç kullanan oyuncu iki saniyelik bir hız takviyesi kazanır. İki saniyenin sonunda oyuncu bitap düşmüştür. Bitap düşen oyuncu dört saniye boyunca normal hızının yüzde altmışı kadar bir hızla hareket edebilir. Dört saniyenin sonunda oyuncunun hızı normale döner ve özel güç için bekleme süresi başlar. Bekleme süresi boyunca oyunca özel güç kullanamaz. Ancak bekleme süresi sona erince tekrardan özel güç kullanabilir. Özel güç, tehlikeli durumlardan kaçmaya yarar sağlayabildiği gibi av kovalamak için de faydalı olacaktır. Ancak özel güç kullanıldıktan sonra oyuncu hızı düştüğü için bir süreliğine savunmasız kalacaktır. Bu yönüyle özel güç faydalı olabileceği gibi doğru kullanılmazsa zararlı da olabilir. İki boyutlu hareket ve özel güç hesaba katıldığında oyuncu iki harekette bulunabilir. Ancak iki boyutlu hareket iki değişken ile ifade edilebilmektedir. Hareket yönünün x ekseni ve y ekseni iki ayrı değişken olacaktır. Böylece yapay oyuncunun sinir ağında üç çıktı olacaktır.

Cell Rush oyununda Bölüm 4.2'de bahsedildiği gibi özel bir algoritma ile çevre dünyada algılanan objeler üç farklı vektöre indirgenir. (Ayrıştırma algoritması) Bu vektörlerin her biri ikişer adet girdi değişkenini oluşturacaktır. Oyuncunun hızı, hacmi, özel gücün bekleme süresi ve oyuncunun açlık değeri de diğer girdi değişkenleridir.

Oyuncunun hacmi ne kadar büyükse o kadar yavaş ilerleyecektir. Ayrıca oyuncunun hacmi etraftaki objelerin yemek mi tehdit mi olduğunu anlaması için de kullanılır. Oyuncu hızı da kaçmak veya kovalamak için gerekli değişkenlerdendir. Oyuncu belirli bir süre bir şey yemezse açlık seviyesi kritik düzeye ulaşacak ve oyuncu can kaybetmeye başlayacaktır. Bu yüzden oyuncunun daha girişken olup gerekirse risk alması gerekir.

10 adet girdi ve 3 adet çıktı değişkeniyle Cell Rush yapay oyuncusunun sinir ağı 10x8x4x3 şeklinde bir yapay sinir ağı olacaktır. Cell Rush yapay oyuncusunun sinir ağının toplamda 134 ağırlığı bulunur. Bu değer diğer iki yapay oyuncudan daha fazladır.

Cell Rush oyununda amaç üç dakika içerisinde en fazla puanı almaktır. Eğer oyuncu üç dakika içerisinde açlıktan ölür veya bir rakip/engel tarafından öldürülürse doğma noktalarından birinde tekrar doğar ve oyuna devam eder. Bu yüzden tam olarak yanma durumu yoktur. Çünkü oyuncu ölünce oyun döngüsü bitmez. Cell Rush Oyununda, bu sebeple başarı kriteri olarak önceki iki oyunda olduğu gibi hayatta kalma kriter alınamaz. Cell Rush yapay oyuncusu için başarı kriteri belirlenen skora (1200) ulaşmaktır.

Başarı kriteri belirlenen skora ulaşmak olduğu için uygunluk değeri de oyuncunun aldığı skor olacaktır. Bu skora ulaşmak için oyuncu çok sık yanmamalı ve hızlı bir şekilde (dakikada ortalama 400 puanlık) puan toplamalıdır. Oyuncu açlıktan ötürü yandığında 40 puan, diğer sebeplerden ötürü yandığında 20 puan kaybeder. Her bir yiyecek besleyicilik değeri kadar puan getirdiğinden sürekli plankton yemek yeterli puan toplanamamasına neden olur. Bu sebeple oyuncu bazı durumlarda diğer oyuncuları (hücreleri) avlayıp sitoplazma parçası da yemelidir. Sitoplazmalar zamanla çürüyüp yok olmakta ve bu yüzden verdiği puan süre geçtikçe azalmaktadır. Ayrıca oyuncular belirli bir hacme ulaşınca seviye atlamaktadırlar. Her seviye atlama oyuncuya 50 puan kazandırmaktadır. Cell Rush Oyunu için uygunluk fonksiyonu:

$$F = 2\,x\,P + 4\,x\,S + 50\,x\,L$$

Olacaktır. Bu fonksiyonda P = Yenilen plankton toplam besin değeri, S = Yenilen Sitoplazmaların toplam besin değeri, L = Seviye atlama sayısı olacaktır.

Cell Rush oyununda her bir oturum 3 dk sürdüğü için sınama süresi de 3 dakika olacaktır. 3 dakikanın sonunda bireyler değerlendirilir ve YOGO ana döngüsüne devam edilir.

Cell Rush yapay oyuncuları da önceki yapay oyuncular gibi tekdüze (uniform) çaprazlama tekniği ile çaprazlanır. Ancak seçilim teknikleri sıralı seçilim olup, mutasyon oranı %2'tir. Diğer oyunlarda olduğu gibi mutasyon şansı performansa göre değişiklik göstermektedir.

Maksimum iterasyon sayısı 1000 tutulmuş ve 1000 jenerasyon ulaşıldıktan sonra eğer başarı kriteri sağlanmamışsa simülasyon sonlandırılmıştır. Toplamda üç kez eğitim yapılmış ve sadece üçüncü eğitimde istenilen puana ulaşılmıştır.

İlk simülasyon oyuncuların açlık hızının %100 olduğu ve ölüm penaltısının (açlıktan ölme -40 puan, öldürülme -20 puan) %100 olduğu Zorlayıcı modda yapılmıştır. Bu modda yapay oyuncular ancak 510 puana ulaşabilmiştir.

İkinci simülasyon açlık hızının %50, ölüm penaltısının %100 olduğu Mücadele modunda yapılmıştır. Bu simülasyonda oyuncular 925 puana çıkabilmişlerdir.

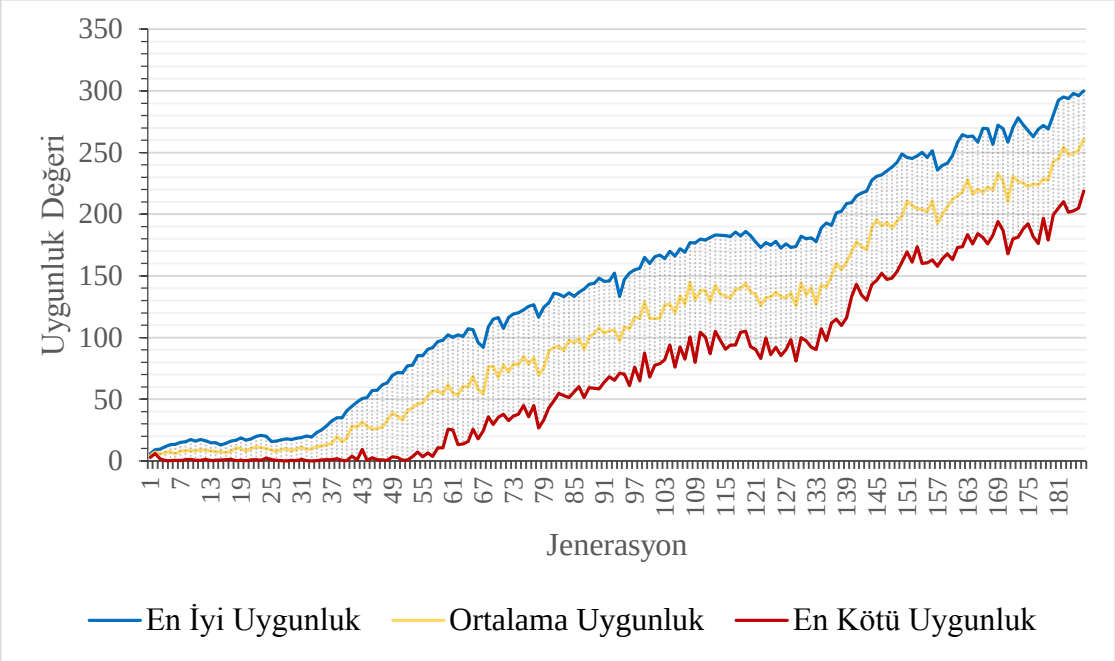
Son olarak üçüncü simülasyonda oyuncuların yapay sinir ağı değiştirilerek [9 x 8 x 4 x 3]'lük bir yapay sinir ağı kullanılmıştır. Çünkü üçüncü simülasyonda kullanılan oyun modunda açlık çıkarılmıştır. Bu yüzden oyuncuların yapay sinir ağına gönderilen açlık değeri ortadan kaldırılmıştır. Ayrıca uygunluk fonksiyonunda seviye atlamada alınan puan 50'den 10'a düşürülmüştür. Bu simülasyon sonucunda 912. Jenerasyonda 1200 puan aşılmış ve hedefe ulaşılmıştır.

5. ARAŞTIRMA BULGULARI

5.1 Flappy Bird Oyununda YOGO Kullanılması

Flappy Bird oyunu için tasarlanan yapay oyuncuların jenerasyonlara bağlı uygunluk değerleri çizelge 5.1.'de verildiği gibi olmuştur. İlk 35 jenerasyonda yapay oyuncular herhangi bir eğitim belirtisi göstermemiş ve bu yüzden mutasyon oranı artmaya devam etmiştir. Mutasyon sonucu oluşan bireylerden birinin önceki jenerasyonlara göre daha iyi performans göstermesiyle jenerasyonların uygunluk değeri artmaya başlamıştır. Bazı noktalarda takılma veya gerileme gösterse de yapay oyuncular gelişmeye devam etmiş ve 188. Jenerasyonda hedef uygunluk puanına (300 puan) ulaşılmıştır. Yapay oyuncuların ilk popüler stratejisi sürekli zıplayarak ekran ortasında kalmak olmuştur. Bu durum ilk iki üç borunun geçilmesine olana sağlasa da dördüncü borudan sonra ekranın üstünde veya altında aralık denk gelmesinden dolayı tüm yapay oyuncuların yanmasına sebep olmuştur. İkinci strateji olarak oyuncular ortada kalmak yerine belirli frekanslarda zıplayarak yay çizmeye başlamıştır. Ancak bu strateji de belirli bir yere kadar işe yarasa da sonrasında yapay oyuncuların yanmasına sebep olmuştur. 120. Jenerasyondan sonrasında oyuncular boruların aralığını izleme stratejisi geliştirmiş ve bu stratejide uzmanlaştıkları vakit hedef puana ulaşmışlardır.

Çizelge 5.1 Flappy Bird oyununda yapay oyuncuların jenerasyonlara göre uygunluk değeri grafiği



5.2 Dino Run Oyununda YOGO Kullanılması

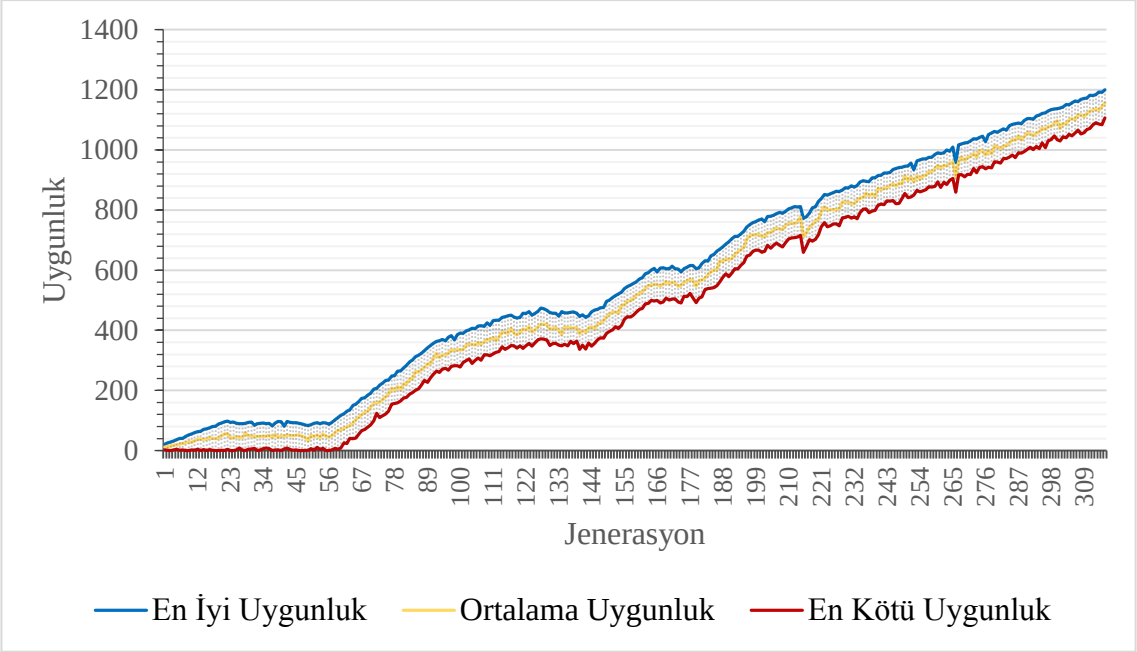
Google Dino Run oyunu için tasarlanan yapay oyuncuların jenerasyonlara bağlı uygunluk değerleri çizelge 5.2. ve 5.3'te verildiği gibi olmuştur. Çizelge 5.2'deki veriler 32 popülasyonu simülasyonun, çizelge 5.3'teki veriler de 64 popülasyonu simülasyonun sonuçlarıdır.

İlk simülasyonda yapay oyuncuların geliştirdiği ilk strateji hiçbir şey yapmamak sadece rastgele anlarda zıplamak veya eğilmektir. Bu strateji yapay oyuncuların, şans eseri on saniye kadar oyunda kalmasını sağlamış olsa da hedefe ulaşmada işe yaramamıştır.

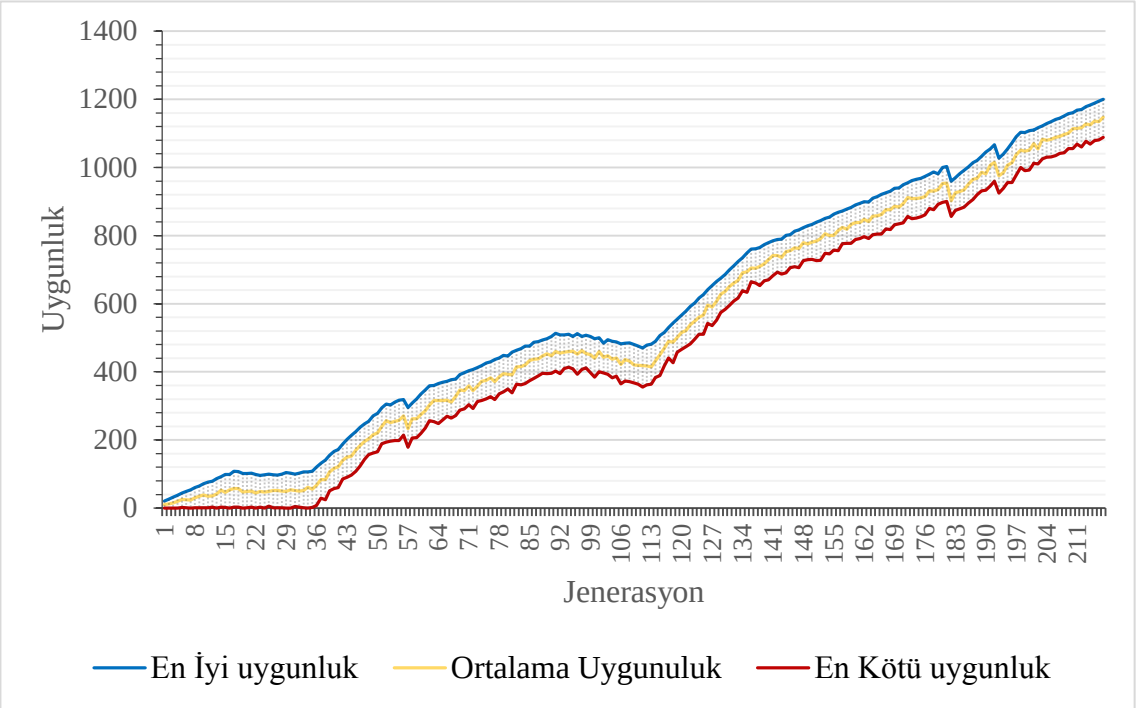
Yapay oyuncular ikinci strateji olarak sürekli zıplamayı hedef almışlardır. Ancak 20. Saniyeden sonra gelen Pterodactyl'ler stratejilerini bozup oyuncuların anmasına sebep olmuştur. Yaklaşık 200. Jenerasyondan sonra yapay oyuncular zıplama ve eğilme hareketlerini doğru kullanmaya başlamış ve son 70 jenerasyonda oyunda uzmanlaşarak hedef puana (1200 puan) ulaşmayı başarmıştır. Hedefe 316. Jenerasyonda ulaşılmıştır.

İkinci simülasyonda 64 büyüklüğünde popülasyon kullanılarak yapılan eğitimler ilk simülasyona göre daha hızlı sonuçlanmış ve yapay oyuncular 217. Jenerasyonda 1200 puana ulaşmıştır. İlk simülasyondaki oyuncuların geliştirdiği stratejiler ikinci simülasyonda da daha hızlı şekilde geliştirilmiştir. İkinci simülasyonda oyuncuların sürekli atlama stratejisi geliştirdiği esnada bazı oyuncular da sürekli eğilme stratejisi geliştirmiş ama Pterodactyl'lerin kaktüslere göre ortaya çıkma oranının düşük olmasından ve Pterodactyl'lerin 20. Saniyeden sonra ortaya çıkmasından dolayı bu strateji sonraki jenerasyonlara aktarılmamıştır.

Çizelge 5.2 Google Dino Run oyununda yapay oyuncuların jenerasyonlara göre uygunluk değeri grafiği (32 popülasyon)



Çizelge 5.3 Google Dino Run oyununda yapay oyuncuların jenerasyonlara göre uygunluk değeri grafiği (64 popülasyon)



5.3 Cell Rush Oyununda YOGO Kullanılması

Cell Rush oyunu için üç farklı simülasyon çalıştırılmıştır. İlk simülasyon zorlayıcı modda, ikinci simülasyon mücadele modunda son simülasyon da sınırsız modda çalıştırılmış olup bu modlar arasındaki fark Çizelge 5.4'teki gibidir.

Zorlayıcı Modunda oyuncu açlık seviyesi %100 hızda artarken, mücadele modunda %50 hızda, sınırsız modunda da %0 hızla artmaktadır. Bu yüzden sınırsız modda oyuncunun yapay sinir ağı düzenlenerek [9,8,4,3] yapılmıştır. Çünkü normalde girdi olarak belirlenen açlık seviyesi değeri sınırsız modda kullanılmayacaktır. Ayrıca sınırsız modda seviye atlama puanı 50'den 10'a indirilmiştir. Bunun sebebi açlık olmayan durumlarda ölüm sayılarının azalması ve daha rahat bir atmosferin oluşması yapay oyuncuların daha kolay seviye atlamasına ve beklenilen şekilde oyunu oynamamalarına rağmen yapay oyuncuların uygunluk değerinin gereğinden fazla artmasına sebep olmasıdır.

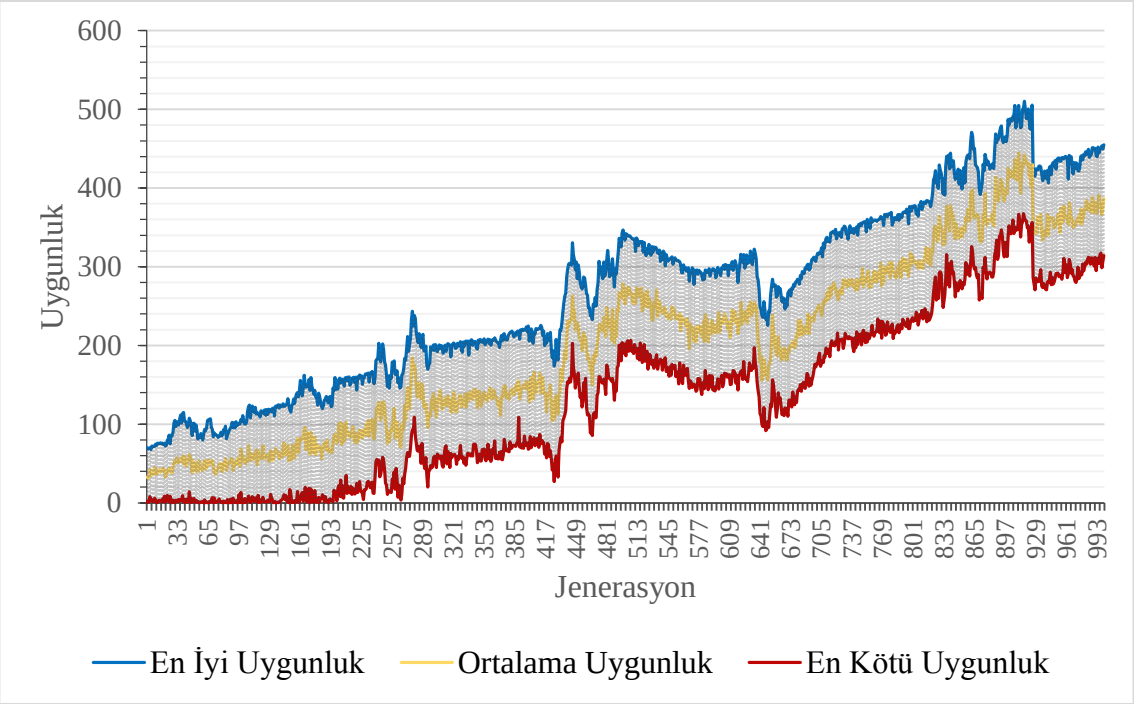
Çizelge 5.4 Cell Rush simülasyonlarında kullanılan oyun modları

MOD	Açlık Seviyesinin Artma Hızı	Açlık Sonucu Yanmanın Penaltısı	Dış Etkenler Sonucu Yanmanın Penaltısı	Ağ Yapısı	Genotip Uzunluğu
Zorlayıcı	%100	40 Puan	20 Puan	[10,8,4,3]	134
Mücadele	%50	40 Puan	20 Puan	[10,8,4,3]	134
Sınırsız	%0	0 Puan	20 Puan	[9,8,4,3]	125

Tüm modlarda oyuncular oyun dünyasının kenarında birbirinden izole şekilde oyuna başlamışlardır. Tuzaklar ve hazır yiyecekler (plankton) haritanın merkezine gittikçe artmaktadır. Oyuncuların hayatta kalabilmesi için harita merkezine doğru yönelmeleri gerekmektedir. Ancak harita merkezine yönelmek, oyuncuların rakiplerle daha hızlı karşılaşmalarına ve tuzakların daha yüksek bir tehdit oluşturmasına sebep olmaktadır. Bir oyuncu kendinden daha büyük başka bir oyuncuya çarpınca veya herhangi bir tuzağa çarpınca can kaybeder. Bu yüzden harita merkezinde ölüm oranlarının daha fazla olması beklenmektedir.

İlk simülasyonda (Zorlayıcı Modda) yapay oyuncular en fazla 510 puana ulaşabilmiş fakat 1000. Jenerasyonda 455 puanla bitirmişlerdir. (Çizelge 5.5) En çok oyunculara puan kaybettiren etken açlıktan dolayı ölmektir. 250. Jenerasyona kadar oyuncular rastgele istikamete gitmişlerdir. Bu oyuncuların harita dışına doğru yol katedenleri açlıktan ölürken merkeze doğru gidenler açlıktan sonra çoğunlukla tuzaklara çarparak ölmüştür. 250. Jenerasyondan sonra yemeğe doğru koşmayı öğrenen yapay oyuncular merkeze yönelmeye başlamış fakat yine öncesinde olduğu gibi çoğunlukla açlıktan ölerek puan kaybetmişlerdir. 417. Jenerasyondan sonra yüksek oranda mutasyon geçiren yeni jenerasyonlar önceki jenerasyonlara göre iyileşme gösterip tuzaklardan kaçınmayı öğrenmişlerdir. Fakat bu durum da merkeze yakın olan oyuncuların birbiri ile savaşmasına sebep olmuş ve 560. Jenerasyondan sonra oyuncuların birbirine ölme oranı

artmıştır. Yine de açlıktan ölme oranı en yüksektir. 640. Jenerasyon etrafında oyuncular hem birbiriyle savaştığı ve hem de açlık çekme sorununa devam ettiği için genel bir performans düşüşü başlamıştır. 673. Jenerasyona kadar düşüş devam etmiş sonrasında yine bir mutasyon sonucu oluşan yeni nesiller duruma uyum sağlamaya başlamıştır. 912. Jenerasyona kadar iyileşen nesiller 918. Jenerasyonda en yüksek puanını (510 puan) almıştır. 929. Jenerasyonda genel bir performans düşüşü gerçekleşmiştir. Bunun sebebi olarak açlık tehdidinin oyunun son dakikasında daha da artmasının, oyuncuların daha çok birbiri ile savaşmasının ve hacmi büyüyen oyuncuların tuzaklardan kaçınmasının daha da zorlaşmasının etkili olduğu düşünülmektedir. 929. Jenerasyondan sonra yapay oyuncular kayda değer bir performans artışı göstermemiş ve 454 puan ile 1000. Jenerasyona ulaşılmıştır.



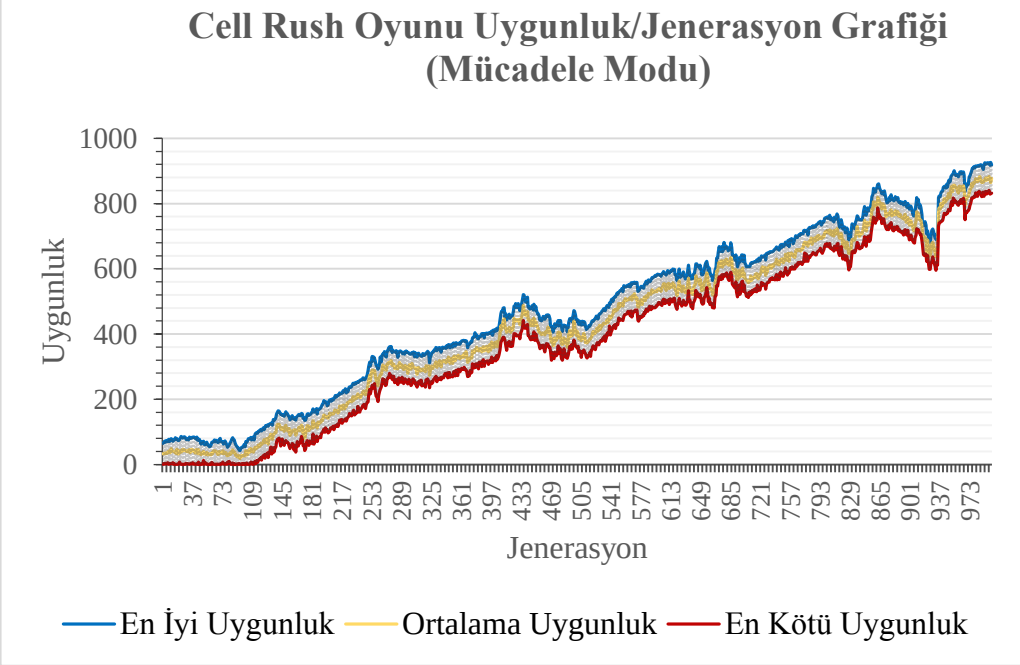
Çizelge 5.5 Cell Rush oyununda yapay oyuncuların jenerasyonlara göre uygunluk değeri grafiği (zorlayıcı mod)

İkinci simülasyonda (Mücadele Modu) açlık hızı yarıya indirilerek daha iyi bir performans elde edilmeye çalışılmıştır. Ancak ilk simülasyondaki gibi hedefe ulaşılamamıştır. (Çizelge 5.6) Yemeğe koşmanın öğrenilmesi ile (Jenerasyon 130 civarı) ve tuzaktan kaçınmanın öğrenilmesi ile (Jenerasyon 270 civarı) yapay

oyuncuların performansı artsa da oyuncuların merkeze yaklaşması ile yine bir kör düğüm ile karşılaşılmıştır. Oyuncuların birbiri ile savaşması artınca oyuncuların duruma ayak uydurmaları yaklaşık 100 jenerasyon sürmüştür. Tehditler sınırına ulaşana kadar (Jenerasyon 930 civarı) ilerleme devam etmiş ama açlık ve tuzaklara, oyuncuların birbiri ile savaşması da eklenince performans düşüşü yaşanmıştır. Mutasyon patlaması ile (Jenerasyon 950 civarında) duruma uyum sağlanmış olsa da hedef uygunluğa ulaşmadan jenerasyon sınırı 1000'e ulaşılmıştır. Yapay oyuncular mücadele modunda en yüksek 925 puan almıştır.

Üçüncü simülasyonda açlık tamamen ortadan kaldırılmış ve yeni bir sinir ağı ile testlere başlanmıştır. Önceki simülasyonda benzer sırasıyla şu kilit adımlar atılmıştır (Çizelge 5.7):

Çizelge 5.6 Cell Rush oyununda yapay oyuncuların jenerasyonlara göre uygunluk değeri grafiği (mücadele modu)



Jenerasyon 930 civarı) ilerleme devam etmiş ama açlık ve tuzaklara, oyuncuların birbiri ile savaşması da eklenince performans düşüşü yaşanmıştır. Mutasyon patlaması ile (Jenerasyon 950 civarında) duruma uyum sağlanmış olsa da hedef uygunluğa

ulaşmadan jenerasyon sınırı 1000'e ulaşılmıştır. Yapay oyuncular mücadele modunda en yüksek 925 puan almıştır.

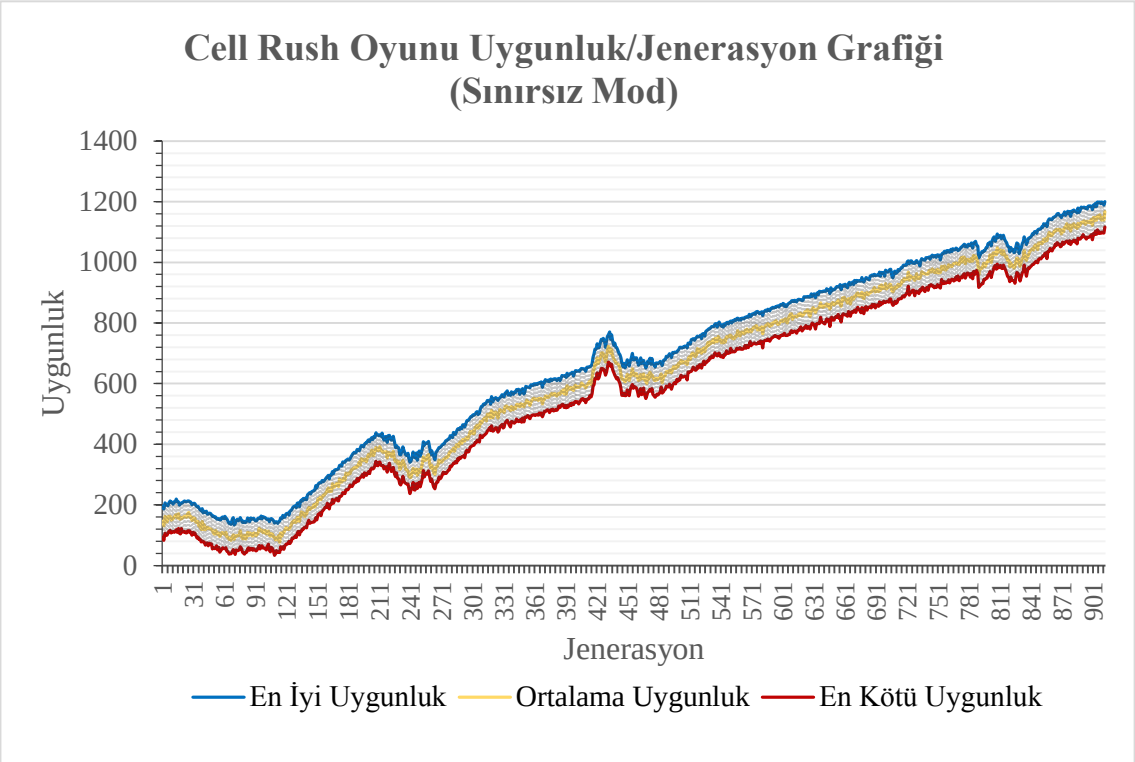
Üçüncü simülasyonda açlık tamamen ortadan kaldırılmış ve yeni bir sinir ağı ile testlere başlanmıştır. Önceki simülasyonda benzer sırasıyla şu kilit adımlar atılmıştır (Çizelge 5.7):

- Yiyeceğe yönelmeyi öğrenme (Jenerasyon 120 civarı)
- Tuzaklara yakalanma (Jenerasyon 210-275 arası)
- Tuzaklardan kaçınmayı öğrenme (Jenerasyon 270 civarı)
- Diğer oyuncularla karşılaşma oranının artmasıyla savaşların başlaması (Jenerasyon 310 civarı)
- Savaşmayı başaran oyuncuların öne çıkması (Jenerasyon 420 civarı)
- Savaşamayan oyuncuların merkezden kaçınarak hayatta kalması ve savaşçı oyuncuların av bulamaması (Jenerasyon 450 civarı)
- Ekosistem dengesiyle birlikte yapay oyuncuların kendini yavaş yavaş geliştirmesi (Jenerasyon 540-780 arası)
- Savaşçı oyuncuların, savaşçı olmayan oyuncuların taktiğine karşı (kaçınma) yeni taktikler geliştirmesi (Jenerasyon 800 civarı)
- Oyuncuların tekrar ekosistem dengesi yakalaması ile kendi performanslarını geliştirmeleri. (Jenerasyon 950 ve sonrası
- Oyuncuların uygunluk hedefi 1200 puana ulaşması (Jenerasyon 912)

Açlığın ortadan kaldırılması Cell Rush oyununu puan toplama yarışına yöneltmiştir. Bu durumda popülasyonun küçük bir kısmı savaşçı olurken büyük kısmı sadece risksiz yiyeceklerin ve bölgelerin peşine gitmiştir. Ancak ne zaman denge sağlansa yiyecek miktarı azalmış ve oyuncuları savaşmaya itmiştir.

Tüm simülasyonlarda yapay oyuncular özel gücün kullanımını kavrayamamıştır. Özel güç ya rastgele kullanılmış ya da her fırsatta kullanılmıştır. Bu da çok iyi performans gösteren bazı oyuncuların beklenmedik şekilde yanmasına sebep olmuştur.

Çizelge 5.7 Cell Rush oyununda yapay oyuncuların jenerasyonlara göre maksimum ve ortalama uygunluk değeri grafiği (64 nüfuslu popülasyon için)



Üçüncü simülasyonda hedef uygunluğa ulaşılsa da verilen jenerasyon limiti içerisinde asıl oyunu oynamayı öğrenen yapay oyuncular yetiştirilememiştir. Eğer 1000 jenerasyondan daha fazla jenerasyon ile simülasyonlar denenmiş olsaydı hedef uygunluk puanına ulaşmak mümkün olacaktı. Ancak 1000 jenerasyonun eğitilmesi gerçek zamanla 3000 dakika sürmektedir. YOGO simülasyonun hızını ayarlayarak 20 kat arttırabilmektedir. Bu durumda eğitim süresi 150 dakikaya inmektedir. Ancak Unity Oyun Motorunun çalışma döngüsünün getirdiği bazı problemlerden dolayı simülasyon hızı 12 katı geçince veri kaybı yaşanmasına veya oyuncuların istenmeyen oyun içi fizik problemiyle karşılaşmasına sebep olmaktadır. Bu sebeplerden dolayı eğitim süresi diğer evrimsel sinir ağı kullanan sistemlere kıyasla daha yavaş sürmektedir.

6. TARTIŞMA VE SONUÇ

Tasarlanan yapay oyuncuların performansına bakılarak YOGO'nun bilgisayar oyunlarına yapay zeka geliştirmek için kullanılabilecek bir sistem olduğu görülmektedir. YOGO'nun eğittiği yapay zekalar ilgili oyunu ortalama bir insan oyuncu seviyesinde oynayabilmekte ve ilgili oyun için istenilen başarı kriterlerini sağlayabilmektedir.

Flappy Bird oyununda bir dakika boyunca yanmayan yapay oyuncular YOGO ile biraz daha eğitilip hiç yanmayacak seviyeye gelebilmekte ve hiçbir insanın ulaşamayacağı düzeyde bir oyuncu halini alabilmektedir.

Google Dino Run oyunu için de YOGO sisteminin eğittiği yapay oyuncular iyi sonuçlar vermektedir. Eğitilen yapay oyuncular günlük skorları aşabilmekte ve en iyi insan oyuncular kadar yüksek performans göstermektedir.

Cell Rush oyununda eğitilen yapay zekalar oyunun kuralları esnetilmeden önce hedef 1200 puana ulaşamamıştır. Bu durum YOGO'nun başarısız olduğu manasına gelmemektedir. Çünkü öncelikle yapay oyuncu için konulan puan hedefi teorik olarak geçilebilir bir hedef olsa da pratikte ulaşılmaz bir hedef olabilir. Bunun iki nedeni vardır: Birincisi Cell Rush oyunu bu proje için geliştirilen ve en iyi oyuncu skorlarının tutulmadığı bir oyundur. Bu yüzden kıyaslanabilecek bir insan oyuncu verisi yoktur. İkinci sebebi de Cell Rush oyununda normal oyun türünde oyuncu sadece çevreye karşı ve alt düzey yapay zekalara karşı yarışmaktadır. Ancak eğitimlerde yapay oyuncular kendisiyle aynı seviyedeki diğer yapay oyunculara karşı mücadele etmek zorunda kalmıştır. Bu da oyunun orijinal zorluğundan daha zor bir ortamda yapay oyuncuların eğitildiği manasına gelir. Yine de YOGO'nun Unity Oyun motorunda eğitilmesinden ve yapay sinir ağı oluşturulurken yapılan ayarların yeterince esnek olmamasından doğan problemlerden ötürü eğitimler çok uzun sürmüş, beklenilen performansın altında bir performans ile eğitimler sonlanmıştır.

Cell Rush oyunu YOGO'nun çok oyunculu oyunlardan ziyade tek oyunculu oyunlar için daha çok uygulanabilir olduğunu göstermektedir. Ancak çalışma yapılırken sunucu kurulup çok oyunculu bir ortamda Cell Rush test edilmediği için YOGO'nun çok oyunculu oyunlara tamamen uygun olmadığı da söylenememektedir.

İleriki çalışmalarda YOGO daha karmaşık oyunlara yapay oyuncu geliştirilerek ve çok oyunculu oyunlarda da kullanılarak daha iyi bir şekilde sınanabilir. Ayrıca YOGO'nun geliştirilmesi için aşağıdaki çözümler göz önüne alınabilir:

1- YOGO için kullanılan yapay sinir ağlarında tüm katmanlarda ve tüm düğümlerde tek bir aktivasyon fonksiyonu kullanılmaktadır. Farklı düğüm ve katmanlarda farklı aktivasyon fonksiyonu kullanmak farklı sonuçlar doğurabilir. Bu yüzden yapay sinir ağı kurulumu için daha esnek bir yapı geliştirilebilir.
2- YOGO genotip oluştururken giriş katmanının da ağırlıklarını göz önüne almaktadır. Normalde yapay sinir ağında giriş katmanının girdi değerleri ile arasında ağırlık değeri yoktur. Bu değerler YOGO'da rastgele atandığından eğitimlerin süresinde artış görülmüştür. Girdi katmanlarının ağırlık değerleri ya ortadan kaldırılmalı ya da kullanıcıya değer atama esnekliği tanınmalıdır.
3- Simülasyon hızı ayarlamadaki sorunlar ortadan kaldırılarak daha hızlı eğitim süresi elde edilebilir.

KAYNAKLAR

Burns E., Laskowski N., Tucci L. 2022. What is artificial intelligence (AI)? https://www.techtarget.com/searchenterpriseai/definition/AI-Artificial-intelligence#:~:text=Artificial%20intelligence%20is%20the%20simulation,speech%20recognition%20and%20machine%20vision. Erişim Tarihi: 16 Şubat 2022

Clune J., Mouret J. B. Ve Lipson. H. 2013. The evolutionary origins of modularity. Proceedings of Royal Society, 280(1755).

Eren A. ve Avuçlu E. 2019. Genetik Algoritmalarda Çaprazlama Yöntemleri. Geleceğin Dünyasında Bilimsel ve Mesleki Çalışmalar (Bilgisayar Mühendisliği) (1-18) Yayıncı: Ekin Basım Yayın Dağıtım

Floreano D., Dürr P. ve Mattiussi C., 2008. "Neuroevolution: From architectures to learning," Evol. Intell., vol. 1, no. 1, 47–62.

Gomez F. J., Burger D. ve Miikkulainen R. A neuro-evolution method for dynamic resource allocation on a chip multiprocessor. International Joint Conference IEEE Neural Networks (IJCNN'01), 4 2001, vol. 4, pp. 2355–2360.

Gunawan, Armanto H., Santoso J., Giovanni D., Kurniawan F., Yudianto R, Steven. 2021. Evolutionary Neural Network for Othello Game. Procedia - Social and Behavioral Sciences, 57, 419-425.

Holland, J. H. (1992). Genetic algorithms. Scientific American, 267(1), 66–73.

Hoover A. K., Szerlip P. A., Norton M. E., Brindle T. A., Merritt Z. ve Stanley. K. O. 2012. Generating a complete multipart musical composition from a single monophonic melody with functional scaffolding. Proceedings of the Third International Conference on Computational Creativity, p. 111.

Nolfi S. ve Floreano D. Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines. Cambridge, MA: MIT Press, 2000.

Risi S., Togelius J. 2017. Neroevolution in Games: State of the Art and Open Challenges. IEEE Transaction on Computional Intelligence and AI in Games, 9 (1), 25-41.

Schrum J., Karpov I. V., and Miikkulainen R., "Human-like combat behaviour via multiobjective neuroevolution," in Believable Bots. New York: Springer, 2012, pp. 119–150.

Stanley K. , B ryant B. ve Miikkulainen R. (2005). Evolving neural network agents in the NERO video game. IEEE 2005 Symposium on Computational Intelligence and Games. April 4-6 2005. CIG'05. Essex University, Colchester, Essex, UK

Şafak I., 2015. İnternetin Yeni Çılgınlığı: Agar.io. https://www.webtekno.com/oyun/internetin-yeni-cilginligi-agar-io-h7483.html Erişim Tarihi: 16 Şubat 2022

The Verge. 2019. How artificial intelligence will revolutionize the way video games are developed and played. https://www.theverge.com/2019/3/6/18222203/video-game-ai-future-procedural-generation-deep-learning. Erişim Tarihi: 13 Şubat 2022.

Togelius J. and Lucas S. M. , "Evolving controllers for simulated car racing," in Proc. 2005 IEEE Congr. Evol. Computation, 2005, vol. 2, pp. 1906–1913.

Warren C. 2014. 28 Days of fame: Th Strange,True Story of Flappy Bird. https://mashable.com/archive/flappy-bird-story. Erişim Tarihi: 18 Şubat 2022

Wikipedia. 2022. Dinosaur Game. https://en.wikipedia.org/wiki/Dinosaur_Game. Erişim Tarihi: 1 Mart 2022

Zanetti S. and Rhalibi A. E., "Machine learning techniques for FPS in Q3," in Proc. 2004 ACM SIGCHI Int. Conf. Advances in Comput. Entertainment Technol. (ACE '04), New York, 2004, pp. 239–244.