

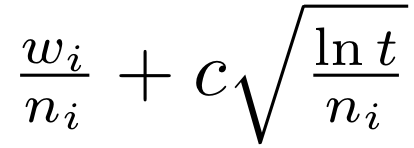
Ders Adı: BLM4510 Yapay Zeka

Katılımcılar: 15011025 Muhammet Çeneli

15011052 Ahmet Görkem Timur

Proje konusu: Monte Carlo Tree ile Tic Tac Toe

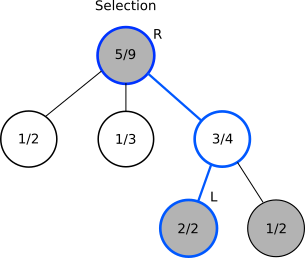
Projede seçilen algoritma olan Monte Carlo Tree ile Tic Tac Toe oyunu yani kısaca MCTS bir sezgisel arama algoritmasıdır. Tree üzerinde node'lar üzerinde aramalarla gerçekleştirilen oyun, her hamle sonrasında skor tablosu tutarak AI'nın ağaç üzerinde kendine en avantajlı durumu hesaplamaktadır. Kendi için en hesaplı durumu hesaplarken aşağıdaki UCB formülü kullanılmaktadır.



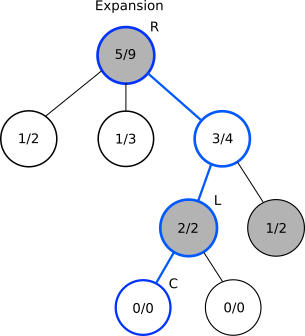
* W: i. hamleden sonra o düğümdeki simüle edilmiş kazanma sayısını gösterir.
* n: i. hareketinden sonra o düğüm için meydana gelen simülasyon sayısını ifade eder.
* c: keşif parametresini temsil eder. Teorik olarak √2’ye eşittir.
* t: i hareketinden sonra meydana gelen toplam simülasyon sayısını ifade eder.

AI’nın karar verme süreci 4 aşamada gerçekleşmektedir. Bunlar selection, expansion, simulation ve back-propagation’dur.

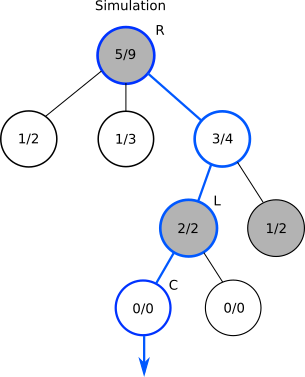
Selection durumunda; kök düğümü, R'dan başlayarak, daha fazla çocuğu olmayan bir düğüme ulaşana kadar, en yüksek UCB değerine sahip ardışık alt L düğümü seçilir.



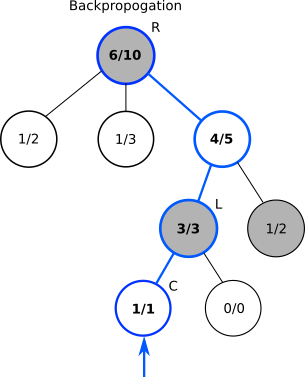
Expansion durumunda; L düğümü oyunu bitiren bir oyun durumu ile sonuçlanmadıkça, expansion aşamasına girilir ve bu oyun durumundan olası tüm sonuçları yaratılır. Sonuç olarak L’nin alt düğümlerinden (C) birini rastgele seçilir.



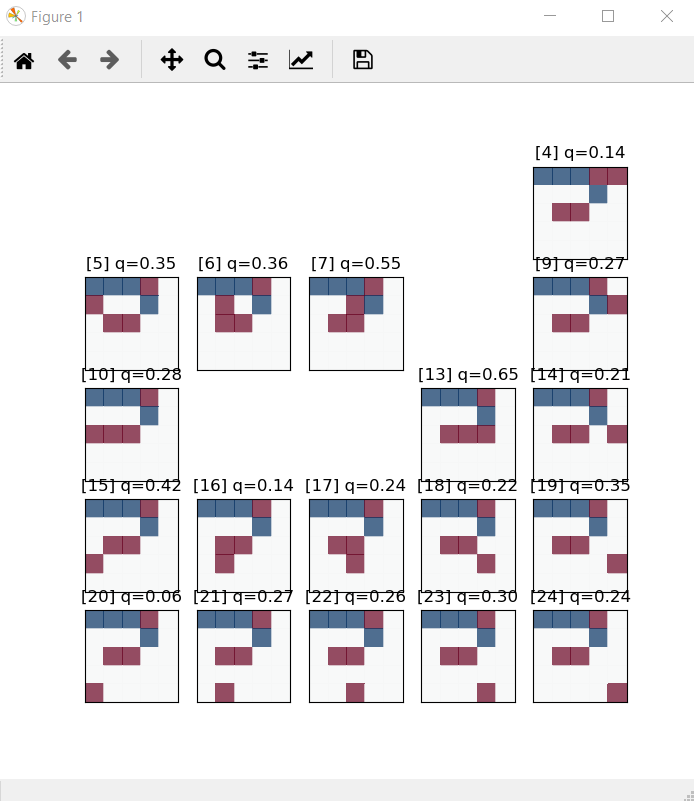
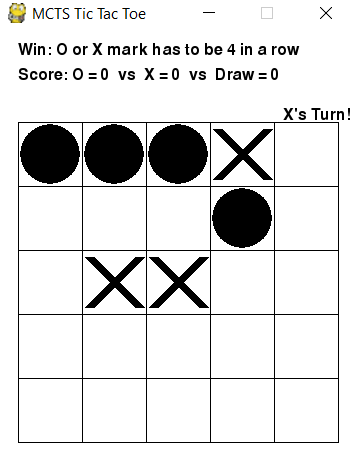
Simulation durumunda; C düğümünden oyun sonu durumuna rastgele bir playout üretilir. (kazan, kaybet, beraberlik)

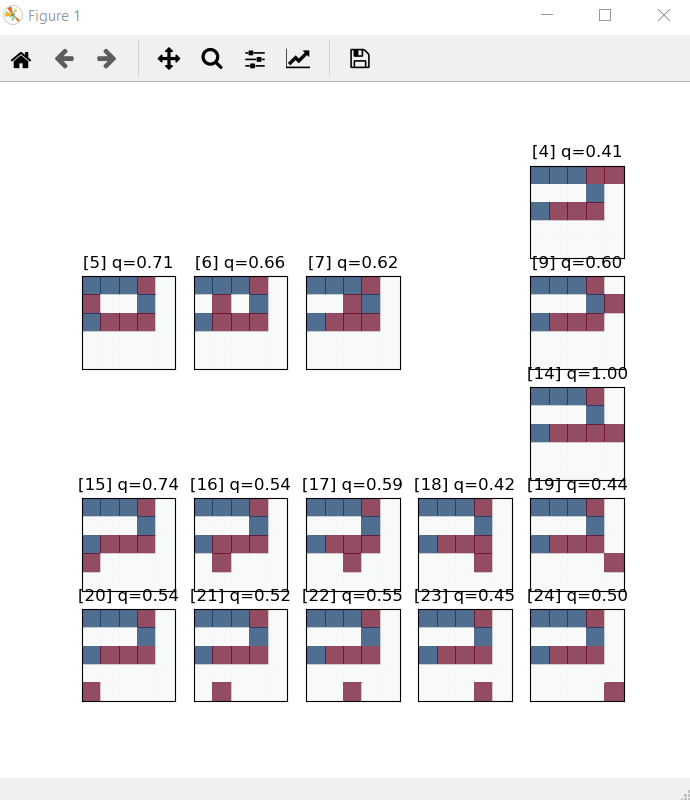
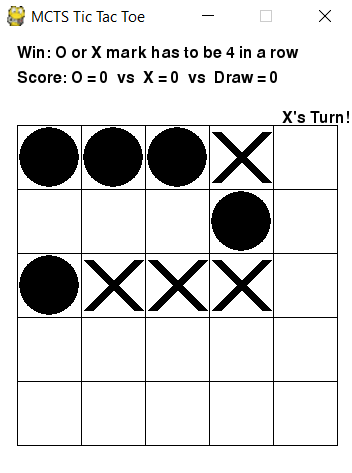


Back-propagation durumunda ise; geriye yayılma sırasında ortaya çıkan oyun durumunu, yeni playout'un kazanma, kayıp veya beraberlik ile sonuçlanıp sonuçlanmadığına dair C'den R'ye düğümlerdeki bilgileri güncellemek için kullanılır.

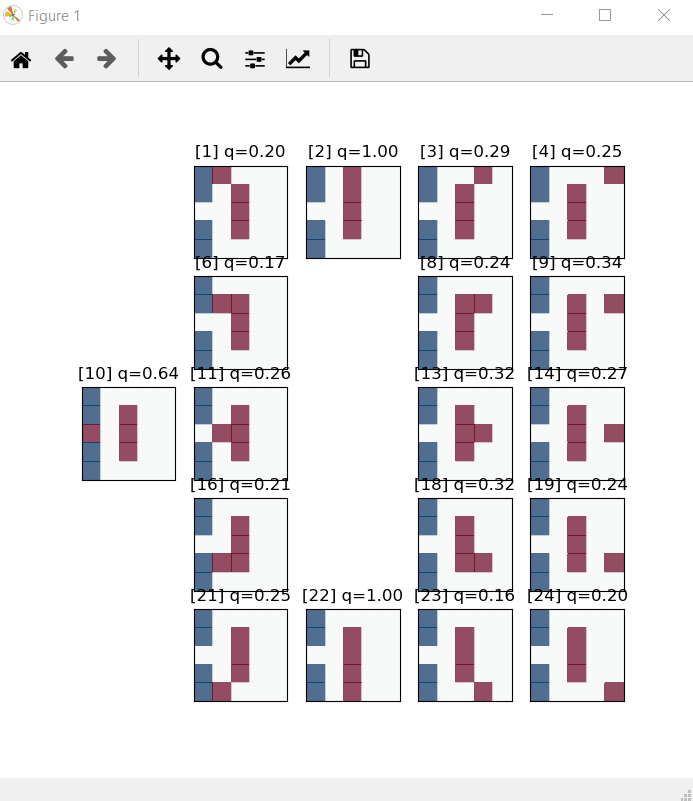
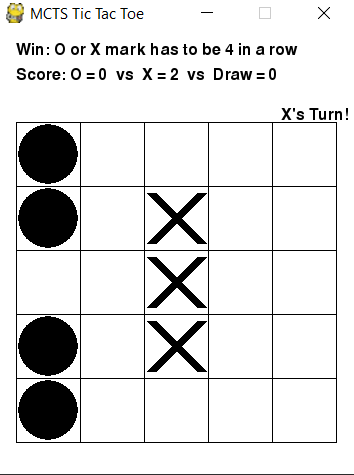


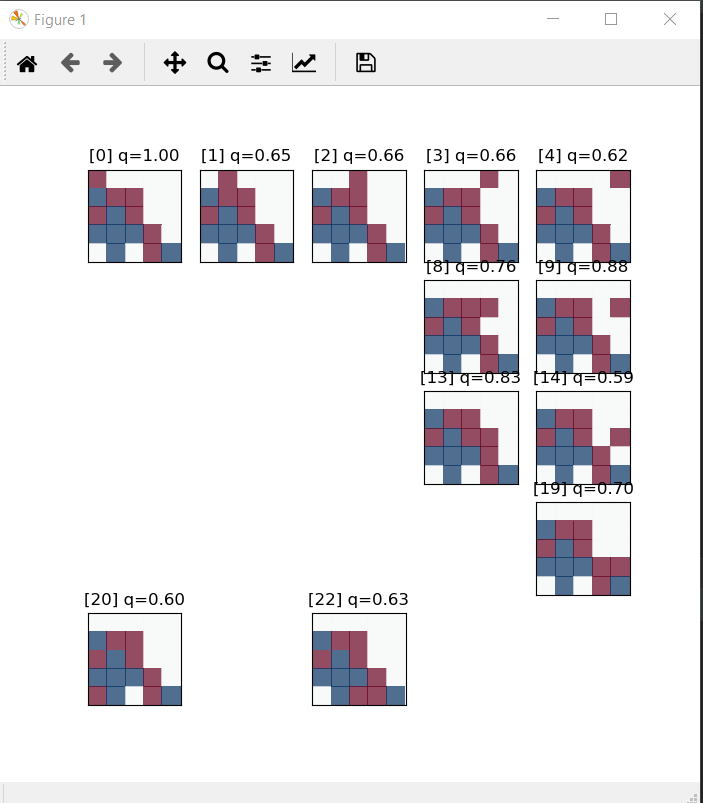
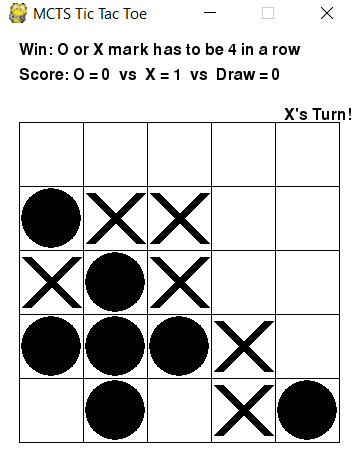
Geliştirme sürecinde MCTS’de yer alan 4 aşama gerçekleştirilmiştir. Çeşitli iterasyonlarda denemeler gerçekleştirilmiştir. İterasyon sayısı düşük tutulduğunda AI’ın düzgün çalışmadığı gözlemlenmiş ve daha yüksek itersyon değerleri kullanılmıştır. Oyun istendiği gibi 5x5 olarak tasarlanmış ve daha dengeli bir oyun ortaya çıkması açısından 4 hamlenin yan yana gelmesiyle kazanç elde edilecek şekilde hazırlanmıştır.



Yapılan her hamle sonucunda AI’ın node’lar üzerinde gezerek olası tahminleri listelenmiş, yaptığı hesaplamalar sonucunda seçtiği en doğru adımı attığı gözlemlenmiştir. Üsteki tabloda görüleceği üzere yaptığı hesaplamaya göre bir sonraki hamlesi q=0. 42 hesaplamasıyla en sol sütun ortadaki satır olacaktır. 

Yine bir üstte görülecek duruma göre AI yaptığı hesaplamarla kazanacağı durumu hesaplamış ve q=1.00 ile en sağ sütun orta satır’a yapacağı hamleyle oyunu kazanmıştır.





Kaynaklar:

<https://www.geeksforgeeks.org/ml-monte-carlo-tree-search-mcts/>

<https://www.baeldung.com/java-monte-carlo-tree-search>

<https://medium.com/swlh/tic-tac-toe-at-the-monte-carlo-a5e0394c7bc2>

<https://en.wikipedia.org/wiki/Monte_Carlo_tree_search>