# RoboCup Arama Kurtarma Benzetim Ortamında Monte Carlo Ağaç Araması Yöntemiyle Planlama

Okan Aşık ve H. Levent Akın
Bilgisayar Mühendisliği Bölümü
Boğaziçi Üniversitesi
İstanbul, Türkiye
Email: okan.asik,akin@boun.edu.tr

***Özetçe* —RoboCup arama kurtarma benzetim ortamı çoklu karar verme problemleri için zorlu bir platform sunmaktadır. Bu çalışmada benzetim ortamında gerçekleşen bir deprem senaryosunda yangınları söndürmek için geliştirilen çoklu planlama yöntemimizi sunuyoruz. Literatürde daha çok kısıtlı eniyileme yöntemleriyle çözülen bu problemi karar verme literatüründeki yöntemlerle çözüyoruz. Problemi Markov Karar Süreci olarak modelleyip Monte Carlo Ağaç Araması yöntemiyle planlama yapıyoruz. Kullandığımız Monte Carlo algoritmasını özgün bir serme algoritması kullanarak iyileştiriyoruz. Markov Karar Süreci benzetim ortamı deneylerinde eniyi sonuçlara çok yakın sonuçlar elde ettiğimizi gösteriyoruz. Ayrıca önerdiğimiz serme algoritmasının başarısının benzetim sayısının artmasına bağlı olarak arttığını gösteriyoruz.**

***Anahtar Kelimeler*—*Çoklu Markov Karar Süreci, Markov Karar Süreci, Monte Carlo Ağaç Araması, RoboCup Arama Kurtarma Benzetim Ortamı***

## I. GİRİŞ

RoboCup Arama Kurtarma Benzetim Ortamı (RAKBO) [1] gerçek haritalar üzerinde deprem sonrası oluşacak yangın, yıkım ve insan davranışlarını benzetmektedir. RAKBO'da dört çeşit etmen bulunmaktadır: sivil, itfaiye, polis ve ambulans. Sivillerin davranışları benzetim ortamı tarafından kontrol edilirken itfaiyeler, polisler ve ambulanslar için kontrolcüler geliştirilmesi hedeflenmektedir. İtfaiyeler oluşan yangınları söndürüp, polisler kapanan yolları açarken, ambulansların görevi yaralıları toplama bölgesine götürmektir. Ana hedef değişik etmenlerin uyum içinde hareket ederek oluşabilecek can ve mal kayıplarını en aza indirecek algoritmalar geliştirmektir.

Literatürdeki çalışmalar ve yarışan takımlar [2] problemi görev dağılımı olarak modellemektedir. Örnek olarak yangın söndürme problemini ele alırsak, harita üzerinde değişik yerlerde çıkan yangınlara hangi itfaiye etmenlerinin müdahale edeceği hesaplanmaktadır. Her itfaiyenin müdahale edebileceği her yangın için bir fayda hesaplanmakta ve kısıtlı eniyileme yöntemleri kullanılarak tüm etmenler için en faydalı eşleşmeler bulunmaktadır. Bu yaklaşımın başarısı fayda fonksiyonun doğruluğuna ve eniyileme yönteminin başarısına bağlıdır.

Markov Karar Süreci (MKS) karar verme problemleri için matematiksel bir model sunmaktadır. MKS problemlerinde, eniyi çözüm politikası değer döngüseli algoritmasıyla hesaplanmaktadır [3]. Değer döngüseli algoritması her durum için alınabilecek eniyi ödülü yinelemeli olarak hesaplamaktadır. Bizim ele aldığımız yangın söndürme problemi birden çok etmenin karar vermesini gerektirdiği için kullandığımız model Çoklu Markov Karar Verme Süreci(ÇMKS) olarak adlandırılmaktadır. Bu modelde eylem seti her etmenin eylemlerinin çapraz çarpımından oluşmaktadır. Eğer modelimizde 3 etmen varsa ve her etmenin iki eylemi varsa toplam eylem sayımız sekiz ($2 \times 2 \times 2 = 8$) olacaktır. Yani eylem sayısı etmen sayısıyla üstel olarak artmaktadır. Eğer durum uzayında etmenlere bağlı faktörler varsa bunlar da modelin karmaşıklığını üstel olarak arttıracaktır.

Literatürde, zor MKS problemleri için geliştirilen yöntemleri iki gruba ayırabiliriz: model yakınsamaları ve yaklaşık algoritmalar. Model yakınsamaları modelin karmaşıklığını azaltacak soyutlamalar yaparak problemi klasik yöntemlerle çözülebilecek şekilde kolaylaştırmayı hedeflemektedir. Yaklaşık algoritmalar da algoritmanın başarısından feragat ederek bir sonuç elde etmeyi hedeflemektedir. Biz çalışmamızda hem model yakınsaması hem de yaklaşık algoritma kullanmaktayız. RAKBO'yu tüm özellikleriyle MKS olarak modellemek mümkün değildir. Bu sebeple, problemi soyutlayarak daha basit bir ÇMKS modeli oluşturuyoruz. Oluşturduğumuz modeli, yaklaşık bir algoritma olan Monte Carlo Ağaç Araması (MCAA) yöntemiyle çözüyoruz.

Monte Carlo Ağaç Araması yöntemi mevcut durumdan başlayarak bir durum-eylem ağacı oluşturmaktadır. Bu ağaç başlangıç durumundan itibaren sürecin bitişine kadar süren benzetimlerle oluşturulmaktadır. En temelde benzetim sırasında toplanan ödüller durum-eylem ikililerinin beklenen değerini örneklemektedir. Belirli bir benzetim sayısına ulaşıldığında en yüksek ödülü veren eylem o durum için seçilmiş olmaktadır.

MCAA algoritması her düğümde seçeceği eylemi üst güven limiti (upper confidence bound-UCB1) [4] algoritmasına göre seçer. Bu algoritmayla düğümlerdeki eylem seçimi çok kollu kumar makinesi (Multi-Armed Bandit) olarak modellenmiş olur. Benzetim sırasında henüz düğüm oluşturulmamış adımlarda serme (rollout) algoritması kullanılmaktadır. Bu algoritma eylem seçimini tamamen rastgele yapmaktadır. Bu çalışmada serme algoritması olarak sezgisel bir algoritma olan *fırsatçı* algoritma kullanılmaktadır.

Yöntemimiz oluşturduğumuz örnek ÇMKS problemlerinde denenmiştir. Bu problemleri küçük seçerek eniyi değerleri elde ediyoruz. Deneylerimize göre yaklaşımımızın %97 oranında eniyi sonuçlara yakın olduğunu gösteriyoruz.

## II. YÖNTEM

Yöntemimiz çevrimiçi bir planlama algoritmasıdır. Mevcut durumu alarak kullanabileceği tüm kaynakları (çalışma süresi) kullanarak bir planlama yapmaktadır. Hesaplanan plan uygulanır ve etmen kendisini yeni bir durumda bulur. İçinde bulunulan yeni durum için süreç tekrarlanır.

### A. RoboCup Arama Kurtarma Benzetim Ortamı Modellemesi

RoboCup Arama Kurtarma Benzetim Ortamı(RAKBO) Çoklu Markov Karar Süreci (ÇMKS) olarak modellenmektedir. Harita üzerindeki her bina ve yol çizge üzerinde birer düğüm olarak gösterilmektedir. Birbirlerine geçişin mümkün olduğu yol ve binalar çizge üzerinde kenarlar ile birbirine bağlanmaktadır.

- Çizge üzerindeki bina düğümlerinin yangın durumları ve etmenlerin çizge üzerinde bulundukları düğümler ÇMKS'nin *durum* uzayını oluşturmaktadır.
- Her etmen bulunduğu düğüm üzerindeki kenarlardan diğer düğümlere geçebilir ya da bulunduğu düğümde durmaya devam edebilir. Tüm etmenlerin eylem kümelerinin çapraz çarpımı *eylem* uzayını oluşturmaktadır.
- Yanan binalar yanmayan binaların $p$ olasılıkla yanmasını sağlamaktadır. Ayrıca etmenlerin bulunduğu binaların yangın durumları söndürülmüş olmaktadır.
- *Ödül* fonksiyonu yanmayan binaların alanlarının tüm harita üzerindeki binaların alanlarının toplamına oranıdır. Yani hiç bina yanmıyorsa ödül $1.0$ ve tüm binalar yanıyorsa $0.0$ olmaktadır.

Bu model RAKBO'daki yangının yayılması ve söndürülmesi problemini soyutlamaktadır. RAKBO'da etmenler harita üzerinde hareket ederken kendilerine verilen maksimum mesafe boyunca hareket edebilirler. Diğer yandan bizim modelimizde her adımda sadece bir düğüm ilerleyebilmekteler. RAKBO'da binalar beş farklı yangın durumunda bulunabilirken, bizim modelimizde binalar sadece iki durumda (*yangın*, *sönmüş*) bulunabilirler. Bizim modelimizde yangın sadece başlangıçta yanan binalardan yayılmaktadır. Bu soyutlamalarla modelin karmaşıklığını azaltırken az bir performans kaybı elde etmeyi hedefliyoruz.

### B. Monte Carlo Ağaç Araması Yöntemi

Monte Carlo Ağaç Araması (MCAA) algoritması yaklaşık bir planlama algoritmasıdır. Algoritma etmenin bulunduğu durumdaki eylemleri bir çok kere benzeterek, her eylemden elde edilebilecek beklenen ödülü kestirmeyi hedeflemektedir. UCT [4] algoritması, arama ağacı oluşturulurken eylem seçiminde keşif ve istismarı dengelemektedir.

En başta arama ağacında başlangıç durumunu gösteren kök düğüm bulunur. Kök düğümden itibaren başlayıp sürecin sonlanmasına kadar süren her benzetim, arama ağacına yeni bir düğüm eklemektedir. Benzetim sonunda alınan ödüller hangi düğümler takip edilerek alındıysa o düğümlerden beklenen ödül değerleri güncellenir. Her düğüm kaç kere ziyaret edildiğini, hangi eylemin kaç kere seçildiğini ve bu düğümden itibaren yapılan benzetimlerden alınan ödülleri saklamaktadır.

**Algoritma** MCAA
**repeat**
  arama($durum$, 0)
**until** Bitiş
**return** Eniyi($durum$, 0)

Şekil 1: MCAA Algoritması [4]

**Algoritma** arama
**if** son($durum$) **then**
  **return** 0
**end if**
**if** yaprak($durum$, $seviye$) **then**
  **return** serim($s$)
**end if**
$action \leftarrow$ eylemSeç($durum$, $seviye$)
($s'$, *ödül*) $\leftarrow$ benzetim($durum$, $eylem$)
$q \leftarrow$ *ödül* + $\gamma$ arama($durum'$, $seviye$ + 1)
değeriGüncelle($durum$, $eylem$, $q$, $seviye$)
**return** $q$

Şekil 2: MCAA Algoritmasının Arama Fonksiyonu [4]

Bu bilgiler sonraki benzetimlerde seçilecek eylemeleri belirlemede kullanılır. Benzetim sırasında uç düğüme gelindiğinde bu düğümden itibaren rastgele eylem seçilerek elde edilebilecek ödüllerin kestirilmesi sağlanır. UCT algoritmasının detayları Şekil 1'de görülmektedir.

UCT algoritması düğümdeki eylem seçimini UCB1 algoritmasıyla yaparak en iyi eylemi seçmemekten kaynaklanan kaybı en aza indirirken mevcut kestirimine göre en iyi olan eylemden faydalanmayı sağlamaktadır. Düğümlerdeki durum-eylem ikililerinin değerleri şu formülle hesaplanmaktadır:

$$Q(s,a) = ortalamaOdul + C\sqrt{\frac{\ln ns}{na}} \qquad (1)$$

$ns$ düğümün ziyaret edilme sayısını, $na$ $a$ eyleminin seçilme sayısını ve $C$ keşif katsayısını göstermektedir. $ortalamaOdul$ bu düğümden itibaren toplanan ortalama ödülü göstermektedir ve ifadenin istismar kısmını oluşturmaktadır. Şekil 2'deki *eylemSeç* fonksiyonu Denklem 1 için en büyük değeri veren eylemi seçer.

Şekil 2'de *son* fonksiyonu sürecin bitmesini kontrol eder, *yaprak* bulunulan düğümün yaprak olmasını ve *benzetim* fonksiyonu verilen durum ve eylemi benzeterek bir sonraki durum($s'$) ve alınan ödülü hesaplar. *serim* fonksiyonu, *serim* algoritması kullanılarak benzetilen süreçten elde edilen toplam ödülü hesaplar. *değeriGüncelle* fonksiyonu da bulunulan düğümdeki verileri günceller.

Literatürde MCAA yöntemlerini iyileştirmek için bir çok yöntem öne sürülmüştür [5]. Bu yöntemlerden en çok kullanılanı, uç düğümden sonra kullanılan rastgele eylem seçimi yapan serme algoritmasını daha iyi bir algoritma ile değiştirmektir. Daha önceki çalışmamızda [6] bu problem için fırsatçı algoritmaların iyi sonuçlar verdiğini göstermiştik.

MCAA algoritmasının varsayılan serme algoritmasını fırsatçı algoritma ile değiştirerek algoritmanın performansının en az fırsatçı algoritma kadar iyi olmasını sağlıyoruz. Kullanılan fırsatçı algoritma, her etmen için en yakın yanan binayı seçmeyi sağlamaktadır. Seçilen binalar için bir patika planlaması yapılır. Daha sonra etmen eylemini patika üzerinde ilerleyecek şekilde seçer. Bu yönteme *Fırsatçı-UCT* algoritması adını veriyoruz.

### C. Bencil Model Yakınsamaları

Markov Karar Süreci (MKS) karar verme problemleri için matematiksel bir model sunmaktadır. MKS dört öğeden $\langle S, A, T, R\rangle$ oluşmaktadır: $S$ durum kümesini, $A$ eylem kümesini, $T$ geçiş fonksiyonu, ve $R$ ödül fonksiyonu göstermektedir. Denklem 2'de görüldüğü gibi her durum-eylem ikilileri için beklenen değer yinelemeli olarak hesaplanmaktadır.

$$Q(s,a) = R(s,a) + \gamma \sum_{s' \in S} T(s,a,s')V(s') \tag{2}$$

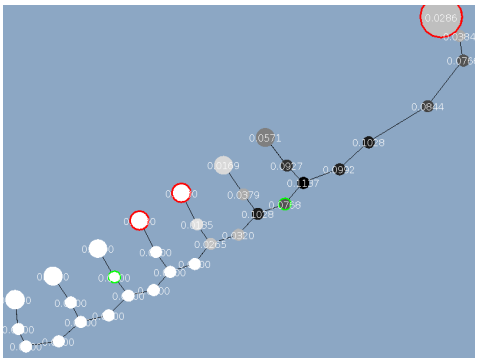
$$V(s) = \max_{a \in A} Q(s,a) \tag{3}$$

Denklemde $s$ durumu, $s'$ bir sonraki durumu, $a$ eylemi ve $\gamma$ gelecek indirimi faktörünü göstermektedir. Bu formülün hesaplanabilirliği durum uzayının $|S|$ büyüklüğüne ve eylem uzayının $|A|$ büyüklüğüne bağlıdır. Sabit ufuklu MKS'ler için (sabit adım süren karar süreçleri) Denklem 2, ufuk adım sayısı kadar hesaplanmaktadır.

Çoklu Markov Karar Süreci (ÇMKS), MKS'ye etmen sayısının eklenmesi ve eylem kümesinin tüm etmenlerin eylem kümelerinin çapraz çarpımı olacak şekilde değiştirilmesiyle oluşmaktadır. Dolayısıyla ÇMKS problemlerinin zorluğu etmen sayısıyla üstel olarak artmaktadır. Bencil model yakınsamasıyla etmen sayısına bağlı karmaşıklığı azaltmak mümkün olmaktadır. Bu yöntemle her etmen evrende sadece kendisi varmış gibi planlama yapmakta ama gelecek indirimi faktörünü diğer etmenlerin toplam etkisine göre hesaplamaktadır.

Yangın söndürme probleminde amaç etmenlerin birbirlerini etkilemeden belirli bölgedeki yangınları kontrol altında tutmalarıdır. Bu sebeple her etmen diğer etmenlerin davranışlarını tahmin edip bu duruma göre kendi davranışını planlayabilir. Tüm etmenlerin davranışlarını özetlemek için *bulunma ağrılığı*'nı [7] kullanıyoruz. Önce her etmen modelde sadece kendisi varmış gibi *Fırsatçı-UCT* algoritmasıyla planlama yapmaktadır. Bu planlama sonucunda durum-eylem ikililerinin ($Q(s,a)$) beklenen değerleri kestirilmiş olmaktadır. Bu değerleri kullanarak başlangıç durumundan itibaren politika üzerinde ufuk adım sayısı kadar yürünmektedir. Bu yürüyüş sırasında bir durumdayken seçilecek eylemler adım sayısıyla sıcaklığı artan bir Boltzmann dağılımı olarak modellenmekte ve etmenin gelecek durumda bulunma olasılığı olarak Boltzmann dağılımınca hesaplanan değer kullanılmaktadır. *Bulunma ağırlığı* algoritması Şekil 4'de görülmektedir. Hesaplanan *bulunma ağırlığı* Şekil 3'de görüldüğü gibi harita üzerinde gösterilebilmektedir. Bu dağılıma baktığımızda etmenin planının en sağ üst köşedeki yangına müdahale etmek olduğunu görebiliyoruz.

Hesaplanan *bulunma ağırlığı*'nı kullanarak her etmen için MKS modelimizde tekrar planlama yapıyoruz. Her etmen



Şekil 3: Düğümlerin renklerinin koyulukları *bulunma ağırlığını* göstermektedir. Kenarları kırmızı düğümler yangın başlangıç noktalarını ve yeşil kenarlı düğümler etmenlerin bulundukları yerleri göstermektedir.

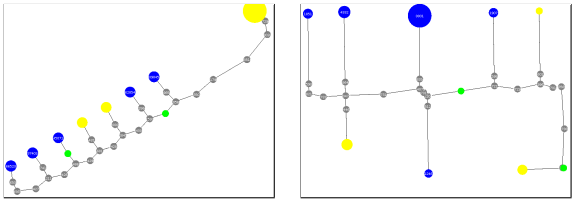
**Algoritma** Bulunma Ağırlığı
**for** etmen $in$ etmenler **do**
  gelecekDurumlar.ekle(mevcutDurum)
  h ← 0
  **repeat**
    **for** durum $in$ gelecekDurum **do**
      qDeğerleri ← UCT.qDeğerleriAl()
      bDeğerleri ← boltzmann(qDeğerleri, h)
      **for** ⟨ a, p, gDurum ⟩ $in$ bDeğerleri **do**
        bulunmaEkle(a, p, gDurum)
        gelecekDurumlar.ekle(gDurum)
      **end for**
    **end for**
    h ← h + 1
  **until** h = ufuk
**end for**

Şekil 4: Bulunma ağırlığı algoritması. $a$ eylemi, $p$ bulunma olsalığını, *gDurum* gelecek durumu göstermektedir.

durumların değerini hesaplarken gelecek durumlardan alacağı değeri diğer etmenlerin bulunma oranına göre azaltıyor. Denklem 4'de görüldüğü gibi gelecek indirimi ($\gamma$) değeri dinamik olarak diğer etmenlerin bulunma olasılığı kullanılarak hesaplanıyor. Amacımız etmeni diğer etmenlerin bulunma olasılığı yüksek olan yerler için cezalandırmak.

$$\begin{aligned} \text{pm}_i(s'_i) &= \sum_{j \neq i} Pr(s_j = s'_i | s) \\ \gamma_i &= (1 - f_i \ \text{pm}_i(s'_i)) \\ Q_i(s_i, a_i) &= R(s_i, a_i) + \sum_{s'_i \in S_i} T(s_i, a_i, s'_i) \Big[ \gamma_i V_i(s'_i) \Big] \end{aligned} \tag{4}$$

$f_i$ ölçeklendirme için kullanılıyor. Bu değeri deneylerimizde en yüksek ödülün, en yüksek değere oranı şeklinde kullandık ve böylece *bulunma ağırlığı*'nın etkisini azalttık.



Şekil 5: Örnek çizge modelleri: gri düğümler yolları, mavi düğümler binaları gösteriyor. Sarı düğümler yanan binaları ve yeşil düğümler de etmenlerin pozisyonlarını gösteriyor.

Tablo I: Algoritmaların adım başına aldığı ortalama ödül

| Fırsatçı | UCT | Fırsatçı-UCT | Bencil-Fırsatçı-UCT | Eniyi |
|---|---|---|---|---|
| 0.625 | 0.635 | **0.686** | 0.629 | 0.712 |
| ± 0.189 | ± 0.165 | ± 0.159 | ± 0.155 | ± 0.147 |
| 87.78% | 89.19% | **96.35%** | 88.34% | 100% |

## III. DENEYLER

Geliştirdiğimiz yöntemi eniyi sonuçlarla karşılaştırabilmek için küçük ÇMKS problemleri oluşturuyoruz. Ayrıca Monte Carlo yöntemi için üst parametre olan benzetim sayısının önerdiğimiz serme algoritmasına etkisini gösteriyoruz.

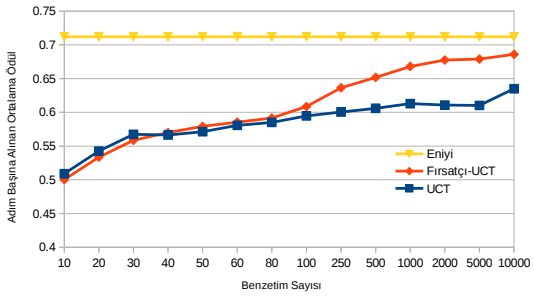
### A. Eniyi Sonuçlarla Karşılaştırma

Elde edilecek sonuçlar hem çizgeye hem de başlangıç durumuna bağlı olduğu için RAKBO'daki beş değişik şehir (*İstanbul*, *Eindhoven*, *Joao*, *Berlin*, ve *Kobe*) haritasından 50 tane harita örnekliyoruz. Her haritada sekiz bina, etmenler için iki tane başlangıç noktası ve üç tane yangın başlangıç noktası seçiyoruz. Böylece 50 tane değişik ÇMKS problemi oluşturuyoruz. Yangının yayılma olasılığını, $p = 0.05$ olarak seçiyoruz ve birbirine 50 metre yakınlıkta bulunan binaları birbirlerinin komşuları olarak belirliyoruz. Örnek modeller Şekil 5'de görülebilir.

Tablo I'de geliştirdiğimiz yöntemin başarısını göstermek için eniyi sonuçlarla karşılaştırıyoruz. Sunulan veriler 100 farklı deney üzerinden hesaplanmış ortalama sonuçlardır. UCT algoritmalarında keşif katsayısı olarak 5 ve benzetim sayısı olarak 10000 kullanılmıştır. *Fırsatçı* algoritması etmenlerin kendilerine en yakın yangını seçmelerini sağlıyor. *Fırsatçı-UCT* algoritması, ÇKMS modelinde planlama yapmaktadır. Eylem seçiminde, tüm etmenlerin eylem kümelerinin çapraz çarpımıyla oluşturduğu eylem kümesini kullanmaktadır. Ayrıca benzetim politikası olarak *fırsatçı* algoritmayı kullanmaktadır. *Bencil-Fırsatçı-UCT* algoritması, Bölüm II-C'de anlatılan ÇMKS problemini MKS'ye dönüştüren yaklaşımı kullanmaktadır. Tablo I'de görüldüğü gibi *Fırsatçı-UCT* yöntemi en yüksek başarıyı elde eden algoritma oluyor ama ne yazık ki eylem sayısı etmen sayısıyla üstel artmaktadır. Bu sebeple çok etmenli problemler için uygun değildir. *Bencil-Fırsatçı-UCT* algoritmasının eniyiye oldukça yakın sonuçlar elde ettiği görülmektedir ve bu algoritmanın karmaşıklığı etmen sayısıyla doğrusal olarak artmaktadır.

### B. Fırsatçı Serme Algoritmasının Etkisi

Rastgele serme algoritmasıyla fırsatçı serme algoritması arasındaki fark Şekil 6'de gösterilmektedir. Bu iki algoritma




Şekil 6: UCT ve Fırsatçı-UCT Algoritmalarının Benzetim Sayısına Göre Performanslarının Değişimi

arasındaki performans farkı benzetim sayısını arttırdığımızda artıyor. Benzetim sayısı az olduğunda kullandığımız serme algoritmasının etkisinin çok az olduğunu görüyoruz.

## IV. SONUÇ

Çoklu karar verme problemi olarak ele alındığında mevcut yöntemlerle makul bir çözümü olmayan RoboCup Arama Kurtarma Problemini model yakınsamaları ve yaklaşık algoritmalar kullanarak çözdük. Kullandığımız Monte Carlo Ağaç Arama algoritmasını fırsatçı serme algoritmasıyla geliştirdik. Çoklu karar verme problemlerinin zorluğunu üstel olarak arttıran etmen sayısının etkisini doğrusal yapabilmek için bencil model yakınsamasını problemimize uyguladık. Gerçekleştirdiğimiz deneylerde yöntemimizin eniyi sonuçlara %96.35 oranında yakın olduğunu gösterdik. Ayrıca önerdiğimiz serme algoritmasının etkisinin kullanılan benzetme sayısına bağlı olduğunu gösterdik.

Gelecekte geliştirdiğimiz yöntemi RoboCup Arama Kurtarma Benzetim Ortamında kullanmayı hedefliyoruz. Modelin gerçek probleme olan yakınlığı başarımızı etkileyeceği için öncelikle yangının yayılmasını, etmenlerin harita üzerinde hareket etmesi ve yangın söndürmesini benzetim ortamında alınan bir çok örneklemle öğrenmeyi planlıyoruz. Daha sonra öğrenilen üretken ÇMKS modelimizi geliştirdiğimiz MCAA yöntemiyle çözmeyi amaçlıyoruz.

## KAYNAKÇA


[1] R. Federation, "Robocup rescue agent simulation," http://roborescue.sourceforge.net/, accessed: 2016-10-05.

[2] R. Sheh, S. Schwertfeger, and A. Visser, "16 years of robocup rescue," *KI-Künstliche Intelligenz*, vol. 30, no. 3-4, pp. 267–277, 2016.

[3] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[4] L. Kocsis and C. Szepesvári, "Bandit based monte-carlo planning," in *Machine Learning: ECML 2006*. Springer, 2006, pp. 282–293.

[5] C. Browne and E. Powley, "A survey of monte carlo tree search methods," *IEEE Transactions on Intelligence and AI in Games*, vol. 4, no. 1, pp. 1–49, 2012. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6145622

[6] O. Aşık and H. L. Akın, "Effective multi-robot spatial task allocation using model approximations," in *RoboCup 2016: Robot Soccer World Cup XX*. Springer Berlin Heidelberg, 2016.

[7] D. Claes, P. Robbel, F. A. Oliehoek, K. Tuyls, D. Hennes, and W. van der Hoek, "Effective Approximations for Multi-Robot Coordination in Spatially Distributed Tasks," in *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems*, 2015, pp. 881–890.