ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

MATRIX NORM BASED-SOLUTION METHODS AND MACHINE LEARNING: STOCHASTIC GAMES AND THEIR APPLICATIONS

Ph.D. THESIS

Murat ÖZKAYA

Department of Mathematical Engineering

Mathematical Engineering Programme

NOVEMBER 2024

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

MATRIX NORM BASED-SOLUTION METHODS AND MACHINE LEARNING: STOCHASTIC GAMES AND THEIR APPLICATIONS

Ph.D. THESIS

Murat ÖZKAYA
(509192246)

Department of Mathematical Engineering

Mathematical Engineering Programme

Thesis Advisor: Assoc. Prof. Dr. Burhaneddin İZGİ

NOVEMBER 2024

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

MATRİS NORM TABANLI ÇÖZÜM YÖNTEMLERİ VE MAKİNE ÖĞRENMESİ: STOKASTİK OYUNLAR VE UYGULAMALARI

DOKTORA TEZİ

Murat ÖZKAYA
(509192246)

Matematik Mühendisliği Anabilim Dalı

Matematik Mühendisliği Programı

Tez Danışmanı: Doç. Dr. Burhaneddin İZGİ

KASIM 2024

Murat ÖZKAYA, a Ph.D. student of ITU Graduate School student ID 509192246, successfully defended the dissertation entitled "MATRIX NORM BASED-SOLUTION METHODS AND MACHINE LEARNING: STOCHASTIC GAMES AND THEIR APPLICATIONS ", which he/she prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Assoc. Prof. Dr. Burhaneddin İZGİ** Istanbul Technical University	.....
Jury Members :	**Assoc. Prof. Dr. Nazım Kemal ÜRE** Istanbul Technical University	.....
	Prof. Dr. Hale GONCE KÖÇKEN Yıldız Technical University	.....
	Prof. Dr. Gülçin Mihriye MUSLU Istanbul Technical University	.....
	Prof. Dr. Fatma İnci ALBAYRAK Yıldız Technical University	.....

Date of Submission : 25 October 2024
Date of Defense : 15 November 2024

To my beloved wife KÜBRA and my family,

FOREWORD

This dissertation is dedicated with the greatest respect and gratitude to my father Faik ÖZKAYA, to my mother Ayten ÖZKAYA and to my brother Emre ÖZKAYA. Many special thanks go out to them for their understanding, supporting, warm behaviors and unconditional believe in me at any time of my life.

I would like to emphasize my heartfelt gratitude to my beloved wife Kübra ÖZKAYA, for her unwavering support in every aspect of my life. Her encouragement and love have been a constant source of strength for me, and I am truly grateful to have her by my side.

I would like to express my heartfelt gratitude to my esteemed advisor Assoc. Prof. Dr. Burhaneddin İZGİ for his invaluable assistance, unwavering encouragement, and patient guidance throughout my Ph.D. journey. His confidence in my abilities has been a tremendous source of motivation, and I am deeply thankful for his support and mentorship.

I would also like to extend my sincere thanks to the members of my doctoral committee, Assoc. Prof. Dr. Nazım Kemal ÜRE from Istanbul Technical University and Prof. Dr. Hale GONCE KÖÇKEN from Yıldız Technical University. Their insightful feedback, support, and valuable suggestions have greatly enriched my research and contributed to the success of my thesis.

This thesis is based on the results of a research project (Grant No: 121E394) conducted within the scope of The Scientific and Technological Research Council of Turkey (TUBITAK) 1001 Project Funding Program. The support of TUBITAK is gratefully acknowledged throughout my Ph.D. education.

November 2024

Murat ÖZKAYA
(Research Assistant)

TABLE OF CONTENTS

Page

ABBREVIATIONS

AI	**:** Artificial Intelligence
CNN	**:** Convolutional Neural Networks
DL	**:** Deep Learning
EMN	**:** Extended Matrix Norm
FNN	**:** Feedforward Neural Networks
ISEMN	**:** Iterative Semi-Extended Matrix Norm
LS-ZS	**:** Large-Scale Zero-Sum
ML	**:** Machine Learning
MN	**:** Matrix Norm
MSE	**:** Mean Squared Error
ReLU	**:** Rectified Linear Unit
RNN	**:** Recurrent Neural Networks
SEMN	**:** Semi-Extended Matrix Norm
SGD	**:** Stochastic Gradient Descent
SHSI	**:** Strong Hybrid Shapley Iteration
SS-ZS	**:** Small-Scale Zero-Sum
WHSI	**:** Weak Hybrid Shapley Iteration

SYMBOLS

$\mathbb{R}$	: Real numbers
$\mathbf{M^c}$	: Column-wise moving matrix
$\mathbf{M^r}$	: Row-wise moving matrix
ϕ	: Activation function
σ	: Sigmoid function
$\hat{y}$	: Predicted value
$\mathbf{w}$	: Weight term
$\mathbf{b}$	: Bias term
∇	: Gradient operator
$\mathbf{v}$	: Game value
$\mathbf{v_{app}}$	: Approximated game value

LIST OF TABLES

Page

LIST OF FIGURES

Page

MATRIX NORM BASED-SOLUTION METHODS AND MACHINE LEARNING: STOCHASTIC GAMES AND THEIR APPLICATIONS

SUMMARY

In this thesis, we comprehensively consider dynamic games including bimatrix games, stochastic games and Markov games. First of all, we develop the matrix norm (MN) method for non-zero sum bimatrix games and comprehensively present the proofs of the extended matrix norm (EMN) method that is extended version of the MN method. These developments provide an improvement for the game value obtained by the matrix norm method. Moreover, a refinement theorem for the boundaries offers more squeezed boundaries for the game value and the theorem also makes the extended matrix norm method suitable for iterative usage. In addition to these, we express some preliminary results associated with the convergence of the matrix norm methods. As an illustration for the implementation of the EMN method, we present some comprehensive applications on different games and also demonstrate the convergence of the method numerically. Additionally, we demonstrate the utilization of the bimatrix games on real-life problems by modeling the risk of infection during Covid-19 and analyze three different country for different stages of first wave of Covid-19 pandemic. We also used repeated bimatrix game to model the people's behavior in a repetitive scenario during the pandemic. Secondly, we consider to eleminate the challenging part of the matrix norm method, which is the usage of the method for the games having more than three strategies in strategy set, by utilizing artifical intelligence. Therefore, we present a new machine learning-driven concept to solve large-scale zero-sum matrix games by using the relations revealed from the offline extended matrix norm method. The proposed neural network architecture take the boundaries produced by the EMN method and the payoff matrix as inputs and provides an estimation for the game value with properly distributed strategy set. The neural network is trained over different games with various sizes up to 50×50. The results show that the neural network can obtain the game values with a relative error less 10% at maximum for even 50×50 sized matrix games and even less for smaller sized matrix games. Thirdly, we investigate stochastic matrix games and their solution methods. We focus on the most common approach, that is Shapley iteration, for solving these games. We begin with boosting Shapley iteration by integrating the extended matrix norm method into the solution process in two different form, weak and strong. Thus, we aim to decrease the necessary number of iterations for Shapley iteration. However, these hybrid methods and original Shapley iteration requires analytic solution in the process so that we introduce two different analytic-solution free methods in order to eliminate this requirement. These numerical methods are establihed on the extended matrix norm method and its iterative form. After the introduction of these methods,

we compare the performance of the proposed methods with Shapley iteration. The results demonstrate that the strong and weak hybrid Shapley iteration works as good as Shapley iteration and some time the the strong hybrid Shapley iteration outperforms all the methods. We also investigate the ranking between these method by using them on relatively larger stochastic matrix games. Fourthly, we consider a single-agent stochastic games, which are also known as Markov reward games. We especially focus on Markov reward games in form of decision tree. We proposed an alternative and holistic matrix norm-based solution method which is distinguished from the existing methods such as value iteration, policy iteration and dynamic programming due to its approach to the solution process. The matrix norm-based method considers all actions and the corresponding stage at once in contrast to the well-known methods that are state and action based methods. The proposed method includes a special transformation of the decision tree into a payoff matrix, the moving matrix concept, 1-norm and ∞- norm. The moving matrix is one of the crucial component of the method since it helps to evaluate the effects of all actions on the stage at once which renders the method holistic. We present a comprehensive solution diagram and an algorithm of the application of the proposed method. We demonstrate the implementation of the method on different games with different state and action sets. In addition to this, we propose a convolutional neural network architecture, that uses the payoff matrix obtained by decision tree transformation and transition matrix as inputs, for solving Markov rewards games. We achieve to compute optimal strategies for the games with relative error less than %3.

MATRİS NORM TABANLI ÇÖZÜM YÖNTEMLERİ VE MAKİNE ÖĞRENMESİ: STOKASTİK OYUNLAR VE UYGULAMALARI

ÖZET

Bu tezde matris oyunları, stokastik oyunlar, Markov oyunları ve bimatris oyunlar gibi çeşitli dinamik oyun türlerini ele aldık. İlk olarak, sıfır toplamlı olmayan bimatris oyunları inceledik. Bu oyun türünün çözümleri, literatürde genellikle çeşitli teoremler kullanılarak denge noktalarının bulunması ya da verilen bimatris oyununun iki farklı matris oyununa dekompoze edilmesi ile elde edilmektedir. Bu tez kapsamında ise İzgi ve Özkaya tarafından 2019 yılında literatüre kazandırılan matris norm (MN) metodunu bimatris oyunlarda kullanmak üzere geliştirmeyi hedefledik. Bu doğrultuda ilk olarak bimatris oyunların iki farklı matris oyununa ayrıştırılarak çözülmesi yaklaşımını benimseyerek bu oyunları için bir notasyon geliştirdik. Daha sonra geliştirdiğimiz notasyonu kullanarak, ilgili matris oyununun oyun değeri için o oyunun oyun değerini de içeren, 1 ve ∞ normuna bağlı alt ve üst sınırlar sunduk ve ispatlarını [1] nolu çalışmadaki kanıtları esas alarak ifade ettik. Ardından, sınırlardaki oyun değerinden kurtulmak üzere bu sınırları geliştirerek sınırları sadece 1 ve ∞ normuna bağlı olacak şekilde revize ettik. Elde ettiğimiz bu aralıktan seçilen geçici oyun değerinin kullanılmasıyla strateji kümesinin maksimum ve minimum elemanları için sırasıyla alt ve üst sınırlar veren bir teorem sunduk. Maksimum ve minimum eleman arasındaki ilişkiyi gösteren ve birinin bilinmesi durumunda diğeri için sınırlar elde etmeyi sağlayan farklı bir teorem ifade ettik. Daha sonra, oyun değeri için sunduğumuz sınırları geliştiren ve farklı matris norm türleri kullanılarak oyun değeri için aralık veren yeni sınırları sunduk. Strateji kümesinin maksimum ve minimum elemanlarının kullanılması ile birlik oyun değeri için elde edilen sınırların çok daha iyi bir hale getirilmesini sağlayan farklı bir teorem ifade ettik ve ispatladık. Ayrıca bu teorem sunulan yöntemin iteratif olarak kullanılmasına olanak sunmuştur. Bu verilen teoremler tez kapsamında sunulan yöntemin matris norm metodundan ayrıştırılmasını ve daha iyi sınırlar elde edilmesi sağlamıştır. Bütün bu teoremlerin sunulması ile birlikte genişletilmiş matris norm yöntemini sunmuş olduk. Genişletilmiş matris norm yönteminin nasıl kullanılacağına ilişkin adımları içeren bir diagram sunarak yöntemi özetledik. Bunlara ek olarak hem matris norm yöntemi hem de genişletilmiş matris norm yöntemi için geçerli olacak şekilde ilgili yöntemlerin yakınsamalarını inceleyerek, teoremler sunduk. Sunmuş olduğumuz yönteme ilişkin ve bimatris oyunların gerçek hayat problemlerine uygulanışını gösteren çeşitli örnekler verdik. Bu kapsamda, ilk olarak cinsiyetler savaşı (Battle of Sexes) adı verilen klasik bir oyunu hem matris norm yöntemi ile hem de genişletilmiş matris norm yöntemi ile çözdük ve bu yöntemlerin sonuç üzerindeki etkilerini ortaya koyduk. Ardından, Taş-Kağıt-Makas oyununu ele alarak matris norm tabanlı bu yöntemlerin yakınsamalarını numerik olarak

inceledik ve sonuçlarını görselleştirerek sunduk. Ayrıca bimatris oyunların gerçek hayat problemlerine uygulanışını küresel bir sağlık sorunu haline gelen Covid-19 pandemisi üzerinden yaptık. Bu örnekte, Türkiye, Güney Kore ve İtalya olmak üzere seçtiğimiz üç farklı ülkenin Covid-19 pandemisinin ilk dalgasındaki karantina uygulamalarını esas alarak oyunu modelledik. İlk dalgayı, Başlangıç-Yayılım-Bitiş şeklinde üç farklı aşamaya ayırdık ve her bir ülkenin her bir aşamadaki davranışını modelleyen oyunlar sunarak denge noktaları hesapladık. Bunlara ek olarak, bireylerin pandemi sürecinde tekrarlı davranışlar sergilemesi durumunda enfekte olma durumlarını incelemek üzere tekrarlı bimatris oyunlar kullarank farklı bir şekilde modelledik ve çözümlerini sunduk. Bunun sonucunda, Türkiye, Güney Kore ve İtalya'nın pandemi boyunca karantinadan nasıl faydalandıklarını gösterdik.

Tezin ikinci bölümünde ise bir önceki bölümde sunulan genişletilmiş matris norm metodunun büyük boyutlu matris oyunlarının çözülmesinde kullanılırken ortaya çıkan strateji kümesinin elemanlarının dağıtılması sorununu ele aldık. Matris norm tabanlı yöntemler 2×2 ve 3×3 boyutlu matris oyunları icin gerek oyun değerinin elde edilmesi gerekse uygun strateji kümesinin oluşturulması acısından rahatlıkla kullanılabilirken, oyun boyutu büyüdükçe strateji kümesinin elemanlarının bulunması ve uygun şekilde dağıtılması zorlaşabilmektedir. Bu zorluğu ortadan kaldırmak amacıyla genişletilmiş matris norm metodu yapay zeka ile desteklenerek strateji kümesinin elemanlarının optimal şekilde dağıtılması ve strateji kümesinin en uygun şekilde oluşturulması hedeflenmiştir. Bunun doğal bir sonucu olarak da büyük boyutu oyunların oyun değerleri daha iyi tahmin edilebilir bir hale gelmiştir. Bu doğrultuda ilk olarak 5×5, 10×10 ve 50×50 boyutlarına sahip, miktarları 500-500000 arasında değişen, getir matrislerini, genişletilmiş matris norm metodu ile elde edilen sınır değerlerini ve gerçek çözüme ilişkin değerleri içeren çeşitli veri kümeleri oluşturulmuştur. Oluşturulan veri kümelerinin %80'nini geliştirilecek olan yapay sinir ağını eğitmek ve kalan %20'si ise test amacıyla kullanılmıştır. Ardından, strateji kümesinin elemanlarını ve dağılımları kestirmek üzere bir yapay sinir ağı modeli sunulmuştur. Geliştirilen bu yapay sinir ağı, getiri matrisini ve genişletilmiş matris norm metodu ile elde edilen p_{max} ve p_{min} sınır değerleri ve oyunun getiri matrisini ve yaklaşık oyun değerini girdi olarak alarak strateji kümesinin eksik elemanları tahmin edip, uygun sıralamada sunacak şekilde tasarlanmıştır. Daha sonra ise yapay sinir ağları yardımı ile elde edilen bu strateji kümesi kullanılarak yaklaşık oyun değeri hesaplanmış ve gerçek oyun değeri ile karşılaştırılmıştır. Elde edilen sonuçlar incelendiğinde 5×5 boyutlu matris oyunları için hata miktarı %4.9, 10×10 boyutlu oyunlar için %5.5 ve 50×50 oyunlar için %9.9 olarak hesaplanmıştır. Hatalara ilişkin elde edilen sonuç görselleştirilerek daha açık bir şekilde sunulmuştur.

Üçüncü bölümde stokastik matris oyunları ve bu oyunların literatürde en çok kabul gören çözüm yöntemi olan Shapley iterasyonunu ele alınmıştır. Bu kapsamda incelenen stokastik matris oyunlarının analitik çözümlerinin oyunun matris boyutu 2×2 iken bile kuadratik denklem çözümünü gerektireceği tespit edilmiştir. Bu nedenle büyük boyutlu stokastik matris içeren oyunları çözümleri için numerik yöntemlerin kullanılması gerekmektedir. Literatürde ise bu tür oyunların numerik çözümlerinin yapılabilmesini sağlayan en bilinen yöntemin Shapley iterasyonu olduğu görülmüştür. Bu çözüm yöntemi oyuna herhangi bir başlangıç değeri vererek iterasyonu başlatıp, oluşan matris oyununu geleneksel yöntemler kullanarak analitik olarak çözmeyi esas almaktadır. Hassasiyet değeri yüksek çözümler elde edilmesini gerektiren oyunlar için ise bu yöntem kullanıldığında iterasyon sayısı ile istenen hassasiyet değerine

bağlı olarak artmaktadır. Bu nedenle ilk olarak bu iterasyon miktarının azaltılmasını sağlayabilecek ve Shapley iterasyonu ile genişletilmiş matris norm metodu iki farklı şekilde birleştirilerek hibrit yöntemlerin elde edilmesi hedeflenmiştir. Bu doğrultuda, Shapley iterasyonu genişletilmiş matris norm yönteminin ilk çıktısını kullanacak şekilde bir araya getirilerek zayıf hibrit Shapley iterasyonu elde edilmiştir. Daha sonra Shapley iterasyonu genişletilmiş matris norm yönteminin iteratif versiyonu ile birleştirilip güçlü hibrit Shapley iterasyonu oluşturulmuştur ve sunulan bu hibrit yöntemlerin ispatları detaylı bir şekilde yapılmıştır. Sunulan hibrit yöntemleri nasıl kullanılacağı açıklayan detaylı bir algoritma sunulmuştur. Bu yöntemlerin hibrit olması nedeniyle, yani ara adımlar Shapley iterasyonunun dahil olması sebebiyle analitik çözüme ihtiyaç duymasından dolayı tamamen genişletilmiş matris norm metodunu temel alan yeni iki farklı iteratif yöntem sunulmuştur. Bu yöntemlerden biri genişletilmiş matris norm metodunu direkt olarak kullanırken diğeri bu yöntemi iteratif olarak kullanmaktadır. Bu yöntemleri sunulması ile birlikte stokastik matris oyunlarının çözümü için kullanılabilecek ve analitik çözümün elde edilmesini gerektirmeyecek alternatif iki metot kazandırılmıştır. Daha sonra bu yöntemlerin uygulanışlarını göstermek amacıyla iki farklı örnek incelenmiştir. İlk örnek 2×2 boyutlu stokastik getiri matrisine sahipken diğeri 3×3 boyutlu stokastik getiri matrisine sahiptir. Geliştirilen dört yöntemin sonuçları Shapley iterasyonu ile elde edilen sonuçlar kullanarak 10^{-3} tolerans seviyesinde karşılaştırılmıştır. Güçlü hibrit Shapley iterasyonunun 2×2 oyun için yaklaşık %0.3'lük hata miktarıyla, sunulan diğer yöntemlerden çok daha iyi performans gösterdiği gözlemlenirken zayıf hibrit Shapley iterasyon yaklaşık %0.4, iteratif genişletişmiş matris norm iterasyonu %1, genişletilmiş matris norm iterasyonu %1.4 hata miktarı ile güçlü hibrit Shapley iterasyonunu performans sıralamasında takip etmektedir. 3×3 boyutlu stokastik getiri matrisine sahip oyun üzerinde bu yöntemler uyguladığında ilk örnekteki performans sıralamasının değişmediği görülmektedir. Öte yandan, sunulan yöntemlerin performanslarının daha detaylı ve daha büyük oyunlar üzerinden test edilebilmesi için 5×5 ve 10×10 boyutlu stokastik matris oyunları oluşturulmuştur. Oyun boyutu büyüdükçe performans sıralamasının korumasına rağmen güçlü hibrit Shapley iterasyonunun diğer yöntemlere karşı daha fazla üstünlük kurduğunu sonucuna ulaşılmıştır. Fakat kurulan bu üstünlüğe rağmen görece olarak en zayıf performansı veren yöntem olan genişletilmiş matris norm iterasyonunun hata miktarı 10×10 boyutlu oyun için bile %0.5-%1.6 arasında değişmektedir. Elde edilen tüm performans kıyaslamaları sonuçlarının daha açık bir şekilde görülebilmesi için sonuçlar görselleştirerek sunulmuştur.

Dördüncü bölümde ise Markov ödül oyunları ve bu oyunların çözüm yöntemleri detaylı bir şekilde ele alınmıştır. Markov ödül oyunları literatürde genel olarak değer iterasyonu, politika iterasyonu veya dinamik programlama gibi yöntemler kullanılarak çözülmektedir. Bu çözüm yöntemlerinde, oyundaki her bir durum ve aksiyon ayrı ayrı değerlendirildiği diğer bir deyişle durum-aksiyon tabanlı olarak yapıldığı gözlenmiştir. Bu nedenle çözüm süreçlerinde sürekli olarak güncelleme ile yeniden hesaplama yapılmasını gerektiği görülmüş ve bu durumun özellikle değer ve politika iterasyonu yöntemleri için daha öne çıktığı tespit edilmiştir. Dinamik programlama yönteminde de benzer bir durum söz konusu olmasına rağmen bu yöntem farklı problemler için farklı sistemler oluşturulmasını gerektirdiği için bir daha farklı bir kullanım dezavantajı oluşturmaktadır. Bu gibi durumlar göz önüne aldığında literatürde Markov ödül oyunlarının, ilgili aşamanın bütün durumlarını ve aksiyonları

tek seferde değerlendiren ve gerekli hesaplamaları bütünsel olarak yapmaya imkan sağlayan bir yöntemin olmaması göze çarpmaktadır. Bu nedenle bu çalışmada matris normları tabanlı holistik yani bütünsel bir yaklaşım sağlayan, aşama-aksiyonları bir arada değerlendirmeye yarayan alternatif bir yöntem sunulmuştur. Geliştirilen matris norm tabanlı holistik yöntem ilk olarak oyunun karar ağacının özel bir yaklaşımla her bir aşama için getiri matrisine dönüştürülmesi işlemi ile başlamaktadır. Daha sonra literatürde hareketli matris olarak bilinen bir matris türü kullanılarak oyunun geçiş matrisi ve oyun ağacından elde edilen getiri matrisi birleştirilerek ilgili aşamanın tüm durumları için ilgili ağırlıklandırılmış ödüllerden oluşan bir matris oluşturulmaktadır. Ardından hareketli matrisin her bir sütununun ∞-normu kullanılarak ve uygun işlem aşamaları takip edilerek farklı bir matris elde edilmektedir. Oluşturulan bu matris kullanılarak bir aşamanın tüm durumları için stratejileri ilgili sütunlardaki sıfırlar ile temsil eden, yani optimal stratejileri sunan bir matris elde edilmektedir. Oyunun bir önceki aşamasındaki ödüllerin güncellenebilmesi için 1-normunu içeren bir prosedür kullanılarak, oyunun ödülleri güncellenmekte ve bu durum ilk aşamaya kadar geriye doğru takip edilerek toplam ödül hesaplanabilmektedir. Bu yöntem sayesinde, tüm oyun holistik bir şekilde çözülebilmektedir. Sunulan matris norm tabanlı holistik yöntemin uygulanışı hem çözüm diyagramı şeklinde hem de algoritmik bir şekilde sunulmuştur. Daha sonra 2-aşama ve 2-aksiyon içeren bir Markov ödül oyunu çözüm diyagramı kullanarak çözülmüştür. Buna ek olarak, 3-aşama ve 3-aksiyon içeren farklı bir oyun algoritma kullanılarak detaylı bir şekilde çözülmüştür. Bunlara ek olarak, geliştirilen bu yöntem ile elde edilen getiri matrisini ve oyununun için verilen geçiş matrisini girdi olarak alan ve optimal stratejileri tahmin edip çıktı olarak veren bir evrişimli yapay sinir ağı geliştirilmiştir. Bu yapay sinir ağını eğitmek için 3000 ve 5000 adet 3-aşamalı ve 3-aksiyonlu Markov ödül oyunlarından oluşan iki farklı veri kümesi oluşturulmuştur. Bu veri kümelerinin %80'nini yapay sinir ağını eğitim, %20'sini ise yapay sinir ağını test etmek için kullanılmıştır. 3000 oyun kullanılarak eğitilen yapay sinir ağının hata miktarı yaklaşık olarak %3, 5000 oyun kullanılarak eğitilen yapay sinir ağının hata miktarı yaklaşık olarak %2.8 olarak elde edilmiş ve ilgili sonuçlar görselleştirilmiştir. Bunun sonucunda hem sunulan yöntem kullanılarak hem de bu sürece yapay sinir ağları dahil ederek daha hızlı sonuçlar elde etmeyi başarılmıştır. Son bölümde ise tez boyunca sunulan tüm araştırmalara yönelik sonuçları sunulmuştur.

1. INTRODUCTION

We investigate several types of dynamic games in game theory such as bimatrix games, stochastic matrix games and Markov games with their applications. We develop and prove extended matrix norm method which is an extension of the matrix norm method introduced in [1]. In addition to this, we demonstrate the convergence of the matrix norm methods. Along the theoretical developments, we present the implementation of the extended matrix norm method and demonstrate the convergence results numerically. We also provide an application of repeated games in extensive forms to model Covid-19 pandemic which is a global health crisis. Additionally, we integrate the extended matrix norm method to artificial intelligence to overcome some practical obstacles and utilize the method efficiently for large-scale matrix games. Moreover, we comprehensively investigate stochastic matrix games and their solution methods. Our focus is centered around Shapley iteration so that we improve four different alternative iterative methods for solving these games. Two of them are based on Shapley iteration, which are hybrid methods containing the extended matrix norm method and also particularly requires the analytic-solution of the game. The rest of the methods are analytic-solution free methods and strongly established on the extended matrix norm method. We compare the performance of these methods with Shapley iteration. Furthermore, we present a holistic and alternative solution method based on matrix norm for solving Markov reward games. Moreover, we construct a convolutional neural network architecture for solving Markov games with different number of action and state sets using some notions from the proposed method.

1.1 Game Theory

Game theory investigates the interaction between agents in a competitive environment. The theory traces back to 1920's with Borel's studies [2–5], however, its breakthrough happens with von Neumann's proof of the minimax theorem in 1928 [6]. The formal representation of game theory is introduced with the book The Theory of Games and

Economic Behavior by von Neumann and Morgenstern in 1944 [7]. They present the fundamental elements of the theory such as strategic form of a game, two person zero sum games and its extension to n-players and so on. In 1950, Nash demonstrates an equilibrium concept describes a situation in which no player can benefit by changing their strategy unilaterally, assuming other players keep their strategies unchanged [8,9]. In 1953, Shapley introduced Shapley value as a method to distribute the total payoff among players according to the each player contribution to the total value [10] and he also defines the concept of stochastic games [11]. In 1954, Shapley and Shubik proposed a method for computing the distribution of power with in legislative or committee systems utilizing simple games and weighted majority games from game theory and applied it on the United States Congress [12]. In 1959, Gillies investigates the general solution approach to nonzero sum games in von Neumann's sense [13].

Schelling, in 1960, examines subject such as common interest, conflict between adversaries, negotiations, war in the sense of game theory using real life problems [14]. In 1965, Rufus presents the games involving differential equations where the players have continuous-time decision [15]. In 1967, Harsanyi investigates game with incomplete information that the players have some beliefs about each other's behavior and he also presents concept of Bayesian equilibrium point [16]. In the same year, Shapley examines a direct correspondence between balanced sets of coalitions of a multi-person game and the conditions which determine if the game has a core [17]. In 1968, Lucas provides a counter-example to the fact that every n-person game has a solution as von Neumann and Morgenstern's solution concept by describing a ten-person game that has no solution [18].

In 1973, Maynard Smith and Price use to game theory to explain the evolution of animal behaviour associate with conflict and competition in their seminal paper and present the concept of the evolutionary stable strategy [19]. In 1975, Selten investigates the equilibrium point for games in extensive form and refines the concept of Nash equilibrium by demonstrating that the player's strategies establishes an equilibrium point not only for the whole game but also every subgames of it and presents the concept of subgame perfect equilibrium [20]. In 1976, Conway presents surreal numbers which is an extension of real number to analyze the games with infinite moves or complex structure [21]. In 1982, Berlekamp et al. analyze the games and strategies

mathematically and establish a base for combinatorial game theory [22]. In 1988, Harsanyi and Selten offer rational criteria for choosing a specific uniformly perfect equilibrium point as the solution for any noncooperative game [23]. In 1992, Roth and Sotomayor investigate how game theory can be utilized to explain the behaviour and results in matching markets [24].

In 1995, Aumann and Maschler study the repeated games, where players interact several times and the equilibrium points for these games. They also improve the concept of correlated equilibrium point and present the analysis of how rational players could achieve cooperative outcomes in repeated games [25]. In 2004, Milgrom investigates how bidding tactics and results can be impacted by various auction forms and proposes comprehensive theoretical framework to understands auction [26]. In 2007, Nisan et al. integrate game theory to computer sciences to understand and solve computational problems in game settings, such as designing efficient algorithms for auction mechanisms, analyzing the complexity of equilibrium computations and therefore the field of algorithmic game theory is developed [27]. In 2011, Camerer comprehensively investigates behavioral game theory which integrates ideas from experimental economics, psychology, and behavioral science to better forecast decision-making processes [28]. In 2016, Sanchez-Perez investigates an axiomatic approach to the structure of partition function games, providing a thorough description of linear symmetric solutions by breaking down game sets and payoff vector spaces [29]. In 2018, Clempner presents game theory problem based on Lyapunov theory, examining it in static and dynamic scenarios, and establishing properties of Nash equilibria using the Lyapunov framework [30].

In 2019, İzgi and Özkaya develop a new solution method to two person zero sum games in strategic form using the matrix norm of the payoff matrices and also present an algorithm for the creation of such games [1]. In the same year, Blanc and Rossi aim to show to use a game theoretical technique to illustrate the conclusions about nonlinear second order partial differential equations [31]. Deng et al. improve Dempster-Shafer theory solution to the zero-sum polymatrix games, which are a class of multi-person games, with link uncertainty [32]. Pangallo et al. present a framework to predict the probability of convergence of the learning algorithms they study towards a stable point,

providing a foundational structure to estimate the stability concerns the algorithm may encounter during player's game learning process [33].

In 2020, Liu demonstrates the relationship between the game value of a matrix game and its eigenvalues and eigenvectors [34]. Babajanyan et al. demonstrate the interaction between energy and entropy in statistical mechanics is formalized through axiomatic bargaining theory, in which negative energy and entropy are expressed as utilities of distinct players, providing a game-theoretic approach to the thermalization problem in addition to current physical techniques [35]. In 2021, Malhotra adapts and prove the matrix norm method proposed in [1] for a 2-tuple linguistic matrix games [36]. Yekkehkhany et al. investigate stochastic games and present novel notion of a stochastic superiority best response which result in stochastic superiority equilibrium [37]. Petrosyan et al. develop a novel cooperative-trajectory characteristic function to assess coalition worth, and so introduces a time-consistent dynamic Shapley value imputation for differential network games [38] .

In 2022, Zheng and Shi investigate Stackelberg stochastic differential game with asymmetric noisy observation in which the follower only has access to noisy observations while the leader has complete observation of the state process [39]. In the same year, Hazra and Anjalia comprehensively survey the role of game theory in machine learning by presenting interrelation between deep learning problems and type of games [40]. Broom and Rychtář discuss and present the utilization of game theory on biology including the concept of matrix games, evolutionary games and evolutionary stable strategy [41].

In 2023, İzgi et al. propose the extended version of the matrix norm method in [1] and present necessary proofs in addition to the demonstration of the convergence of the extended matrix norm method [42]. İzgi et al. propose an neural network that boosts the performance of the extended matrix norm method introduced in [42], for large-scale matrix games [43]. Djebara et al. present a novel solution method to a constrained matrix game whose the payoffs and the constraints are LR-fuzzy numbers and R-saddle point equilibrium [44]. Liu and Guo explore classical two-player zero-sum linear-quadratic stochastic differential games with unknown system coefficient matrices, focusing on adaptive strategies [45]. In 2024, Lozovanu and Pickl propose novel algorithmic results on quasi-monotonic

programming techniques, and introduce stochastic positional games that generalize deterministic ones. They also examine latest developments in finite state-space Markov decision problems and Nash equilibria for stochastic games with average and discounted reward payoffs [46]. İzgi et al. state and prove the matrix norm methods proposed in [1, 42] for zero sum fuzzy matrix games by defining new concept of fuzzy matrix norms and their relationships with each other [47]. Shreyas and Vijesh propose a Q-learning algorithm for solving two player zero sum Markov games and their suggested algorithm converges to the optimal value according to game theory with a probability of one whenever the model information is unknown [48]. İzgi et al. proposed hybrid Shapley iteration methods which integrated to extended matrix norm method introduced in [42], for boosting the original Shapley iteration and in addition to these they present novel alternations to Shapley iteration for solving stochastic matrix games [49]. Crönert and Minner address the shortcomings of existing algorithms by proposing a new solution approach for large finite games that combines equilibrium selection theory and sampling techniques to find all equilibria and choose the most likely one [50]. Dey et al. present a novel game theoretical framework for using quantum strategies in entanglement distribution within quantum networks, showing quantum advantages over classical strategies in link fidelity and communication latency [51]. İzgi et al. propose an alternative holistic method for solving Markov reward games using special transformation of decision tree, matrix norms and the concept of moving matrix [52]. This brief literature review summarizes some of the most important theoretical papers and recent developments in game theory. Since its introduction, game theory has garnered significant attention and continues to be a developing field. Researchers such as Nash, Harsanyi, Selten, Shapley, Aumann, Schelling, Roth, Milgrom and Wilson who have contributed significantly to the field, have been awarded the Nobel Prize for their work. The practical applications of game theory will be discussed in the upcoming chapters, in line with the chapter topics.

The following of the thesis is organized as: Extended Matrix Norm Method and Convergence Results, Machine Learning Application for Large-Scale Matrix Games, Hybrid Shapley and Iterative Methods for Stochastic Games, A Holistic Alternative Method for Stochastic Games: Markov Reward Games, and Conclusions.

1.1.1 Matrix games and solution methods

Definition 1.1.1 (Two Person Zero Sum Game [53]) *The strategic form, or normal form, of a two person zero sum game is denoted by a triplet (X, Y, A) where*

1. *X is a nonempty set, the set of strategies of Player I*
2. *Y is a nonempty set, the set of strategies of Player II*
3. *A is a real-valued function defined on $X \times Y$.*

Definition 1.1.2 (Constant/Zero Sum Game [54,55]) *A game is called constant sum if the sum of payoffs equals to a constant for any outcome. If the constant is zero, then the game is called zero sum game.*

Definition 1.1.3 *[53] Let $A \in \mathbb{R}^{m \times n}$ be the bimatrix of the nonzero sum game as in the following form*

$$A = \begin{bmatrix} (a_{11}, b_{11}) & . & (a_{1n}, b_{1n}) \\ . & . & . \\ (a_{n1}, b_{n1}) & . & (a_{mn}, b_{mn}) \end{bmatrix}$$

where the pair (a_{ij}, b_{ij}) represents the payoff of the Player I and Player II, respectively.

Definition 1.1.4 (Mixed Strategy Set [54]) *Let (X, Y, A) be a normal form game, and any set Z. Let $\Pi(Z)$ be the all probability distribution over Z. Then the set of mixed strategies for Player I (or Player II) is $S_I = \Pi(Z_I)$ (or $S_{II} = \Pi(Z_{II})$)*

Definition 1.1.5 (Equalizing Strategy [53]) *A strategy that produces the same average winnings no matter what the opponent does is called an equalizing strategy.*

Definition 1.1.6 (Solution of a Zero Sum Game [56]) *Let A be the payoff matrix of a game and x_i and y_j be the elements of the mixed strategy sets of the players. Then the game value v is calculated as $v = \max_i \sum_j a_{ij} y_j = \max_j \sum_i a_{ij} x_i$ for all i,j.*

These games can be solved by using classical methods such as linear programming methods, finding saddle points or equilibriums and simply equalizing strategy methods. On the other hand, the games in normal form can be numerically solved by using the matrix norm method proposed in [1].

1.1.2 The matrix norm method

In this subsection, we present briefly the theorems and propositions associated with the matrix norm (MN) method proposed in [1, 57]. The MN method is introduced by İzgi and Özkaya in 2019 [1] and comprehensively investigated in [57]. Next definition presents one of the fundamental tool of the matrix norm method.

Definition 1.1.7 *[58] Let $A \in \mathbb{R}^{m\times n}$ be a matrix. Then, 1-norm and ∞-norm of the matrix A are defined as*

- $||A||_1 = \max\limits_{j}\sum\limits_{i}|a_{ij}|$
- $||A||_\infty = \max\limits_{i}\sum\limits_{j}|a_{ij}|$

where $i = 1,2,...,m$ and $j = 1,2,...,n$.

Next definition explains an important notion called the row-wise and column-wise induced matrices of a real-valued matrix for the MN method. These notions are important to state the boundaries especially for relatively larger matrix games.

Definition 1.1.8 *[1] Let $A \in \mathbb{R}^{m\times n}$ be a real valued matrix, and let $||A||_\infty$ be the sum of absolute values of the h^{th} row's entries, then the matrix $B \in R^{(m-1)\times n}$ is obtained by deleting h^{th} row of the matrix A is called a row-wise induced matrix of A. Similarly, let $A \in R^{m\times n}$ be a real valued matrix, and let $||A||_1$ be the sum of absolute values of the s^{th} column's entries, then the matrix $B \in R^{m\times(n-1)}$ is obtained by deleting s^{th} column of the matrix A is called a column-wise induced matrix of A.*

The following theorem provides boundaries for the game value of a matrix game of size $m \times n$.

Theorem 1.1.9 ***(Generalized Main Theorem) [1]*** *Let A be a $m \times n$ payoff matrix and v be the game value for a two person zero sum game. Then,*

$$\text{if } |v| \geq 1, \text{ then } \frac{||B||_\infty}{||A||_\infty} \leq |v| \leq ||A||_1$$
$$\text{if } |v| \leq 1, \text{ and } v \neq 0, \text{ then } \frac{1}{||A||_1} \leq |v| \leq \frac{||A||_\infty}{||B||_\infty},$$

where B is the row-wise induced matrix of A.

Although the MN method seems valid only for positive matrix games, the next proposition demonstrates that the strategy set is invariant under perturbation and that the game value changes only by the amount of the perturbation. Therefore, any matrix game can be converted to a matrix game with positive entries.

Proposition 1.1.10 *[1] Let A be a 2×2 payoff matrix for two person zero sum game and v be a game value, then the value of the perturbated game $A+X$ with all positive entries (or $A+Y$ with all negative) is $v+x$ (or $v+y$) and the mixed strategy set is invariant, where $X(\text{or } Y) \in \mathbb{R}^{2\times 2}$ with all entries x (or y), and $x = |min(A)|$ (or $y = -|max(A)|$).*

The next theorem presents the lower and upper boundaries for the maximum and minimum elements of the strategy set using matrix norms and the dummy game value selected within the interval obtained from Theorem 1.1.9.

Theorem 1.1.11 *[1] Let $A \in \mathbb{R}^{m\times n}$ be the payoff matrix with positive entries. Then, the boundaries for p_{max} and p_{min} which are the greatest and the smallest elements of the mixed strategy set, respectively, are as follows,*

$$p_{max} \geq L \text{ where } L = max\left\{\frac{1-\frac{v}{||A||_1}}{m-1}, \frac{v}{||B||_1}\right\}$$
$$p_{min} \leq U \text{ where } U = min\left\{\frac{1-\frac{v}{||B||_1}}{m-1}, \frac{v}{||A||_1}\right\}$$

where B is the column wise induced matrix of A.

The following theorem displays the relationship between the maximum and minimum elements of the strategy set and explains how to obtain the boundaries of one using only the other.

Theorem 1.1.12 ***(Min-Max Theorem for the Game Strategies) [1]***
Let $A \in \mathbb{R}^{m\times n}$ be the payoff matrix. Then,

$$\frac{1-p_{min}}{m-1} \leq p_{max} \leq 1-(m-1)p_{min} \text{ and}$$
$$1-(m-1)p_{max} \leq p_{min} \leq \frac{1-p_{max}}{m-1}$$

hold where p_{max} and p_{min} are the greatest and the smallest elements of the mixed strategy set, respectively.

1.1.3 Stochastic matrix games and solution methods

The stochastic games were introduced for the first time in the literature by Shapley in 1953, which models a changing environment with the interaction of the players [11]. Then, the discounted stochastic games, nonterminating stochastic games, stochastic differential games, stochastic matrix games, and so on are described to model different types of games [11, 59, 60].

Definition 1.1.13 *[53] Let G be a stochastic game containing a finite collection of positions or states represented by $\{1,2,...,N\}$ where one of these is selected as the specific initial point, and $G(k)$ refers to the game with the initial position k. Each state, k, is represented as a matrix game $A^{(k)} = (a_{ij}^{(k)})$. If the players in state k select rows and columns of the payoff matrix $A^{(k)}$ at the same time, the following occurs. In the beginning, Player II loses his $a_{ij}^{(k)}$ to the Player I. Second, the game ends or moves another state with some probability depending on i, j and k, where the stopping probability is denoted by the game stops is denoted $s_{ij}^{(k)}$ and probability of the movement of the other state is $P_{ij}^{(k)}(l)$. For all i, j and k*

$$s_{ij}^{(k)} + \sum_{l=1}^{N} P_{ij}^{(k)}(l) = 1.$$

Player I aims to maximize the cumulative payoff of its own and Player II targets to minimize its total loss [53]. The game is described in the following form

$$G^{(k)} = \left(a_{ij}^{(k)} + \sum_{l=1}^{N} P_{ij}^{(k)}(l) G^{(l)} \right).$$

The game has a unique game value and it has an optimal strategy for each player for any starting position. In the next theorem, Equation 1.1 guarantees the uniqueness of the game value whenever Equation 1.2 guarantees the existence of the optimal strategy [53].

Theorem 1.1.14 *[11, 53] Each game G^k has a game value $v(k)$. These values are the unique solution of the set of equations, for $k = 1,2,...,N$,*

$$v(k) = val\left(a_{ij}^{(k)} + \sum_{l=1}^{N} P_{ij}^{k}(l)v(l)\right) \tag{1.1}$$

Each player has a stable optimal strategy, wherein, for state k, the most effective mixed strategy is employed, corresponding to the game characterized by a specific matrix

$$A^k(v) = \left(a_{ij}^{(k)} + \sum_{l=1}^{N} P_{ij}^{k}(l)v(l)\right) \tag{1.2}$$

whenever $v = (v(1),...,v(N))$ shows the vector of values.

The following theorem states the Shapley iteration and the iterative solution of a stochastic matrix game.

Theorem 1.1.15 *[11, 53] In order to start the Shapley iteration, a starting vector is selected, as an example $v_0 = (0,...,0)$, the iteration is convergent for the arbitrary initial value. Then, the following iteration is applied to the stochastic matrix game for $n = 0,1,...$*

$$v_{n+1}(k) = val\left(a_{ij}^{(k)} + \sum_{l=1}^{N} P_{ij}^{k}(l)v_n(l)\right)$$

1.1.4 Markov reward games and solution methods

We also consider Markov reward games, which are mathematical models utilizing the components of the Markov decision process and game theory in the context of this thesis,. In a Markov reward game, the players do not only move among states but also gain some rewards depending on the state transitions and the actions. A Markov reward game consists of states, actions, transition probabilities, rewards, and a discount factor depending on the problem [61, 62].

Definition 1.1.16 *[61, 62] The Markov game is a Markov chain containing values. The Markov reward process is a tuple of* (S,P,R,γ)*, where S is the set of finite states, P is the transition matrix,* $P_{SS'} = P[S_t = s]$*, R is the reward function with* $R_s = E[R_{t+1}|S_t = s]$ *and* γ *is the discount factor with* $\gamma \in [0,1]$.

We present the most prevalent methods for solving Markov reward games. To begin, we outline the value iteration method as follows:

a. Value Iteration. *[63] The value function in the Markov process is evaluated by*

$$V(s) = \max_a (R(a,s) + \gamma \sum_{s'} P(s,a,s')V(s')) \tag{1.3}$$

where $V(s)$ *is the reward in the current state,* $R(a,s)$ *is the reward when action a is selected in state s,* $\gamma \in [0,1]$ *is the discount factor and* $V(s')$ *is the reward of the next state.*

The value iteration method is an iterative algorithm that refines the value function until convergence is achieved, thereby identifying the optimal policy. In this context, the value function quantifies the expected cumulative reward associated with each state. Next, we present another common method for solving Markov reward games, called policy iteration.

b. Policy Iteration. *[64] Let* $S = (1,...,n)$ *be the state space, U be finite decision set for all* $i \in S$*. Given state i, the use of decision* $u \in U$ *specifies the transition probability* $p_{ij}(u)$ *to the next stage j, and a cost* $g(i,u)$ *is incurred when a decision u is taken in state i. Future costs are discounted by a factor* $\gamma \in [0,1]$*. A policy of the stochastic control problem is then to minimize/maximize the so-called cost-to-go function* $J_\mu : S \to U$ *over the set of admissible policies P. By fixing the policy* μ*, the solution of the following linear system is*

$$J_\mu(i) = g(i,\mu(i)) + \gamma \sum_{j \in S} p_{ij} J_\mu(j) \tag{1.4}$$

called the policy iteration.

The policy iteration method can be summarized as an approach that begins with an initial policy and iteratively enhances it by evaluating its performance until it converges to the optimal policy. Next, we will explain the dynamic programming approach and provide a general characterization of it.

c. Dynamic Programming. *[65]* Dynamic programming is a useful mathematical method for making a series of connected decisions. It offers a methodical process for figuring out the best mix of choices. The general characterization of a dynamic programming problem is as follows:

- N: The total number of stages in the process.
- A label denoting the current stage, where n takes values from 1 to N.
- s_n: The current state at stage n.
- x_n: The decision variable at stage n.
- x_n^*: The optimal value of x_n given the current state s_n.
- $f_n(s_n, x_n^*)$: The contribution of stages $n, n+1, ..., N$ to the objective function if the system starts in state s_n at stage n, the immediate decision is x_n, and optimal decisions are made thereafter.

$f_n^* = \max_{x_n}\{f_n(s_n,x_n\}$ or $f_n^* = \min_{x_n}\{f_n(s_n,x_n)\}$, where $f_n(s_n,x_n)$ would be presented in terms of s_n, x_n, $f_{n+1}^*(s_{n+1},x_{n+1})$, and probably some measure of the immediate contribution of x_n to the objective function.

Dynamic programming can be explained as an algorithmic technique that allows solving complex problems by breaking them down into smaller and simpler subproblems.

1.1.5 Some mathematical tools

In this section, we present some notions that are crucial for establishing the methods proposed in this thesis.

Definition 1.1.17 *[66, 67] Let $A \in \mathbb{R}^{m\times n}$ be a matrix. The row-wise moving matrix $M^r \subset A$ is defined as,*

$$M_i^r = A[i : i+a-1, :],\ i = 1,2,3,...,m-a+1,\ M^r \in \mathbb{R}^{a\times n}$$

where a denotes the number of row-wise sub-interval. The colon ":" stands for all the elements in the column of a matrix.

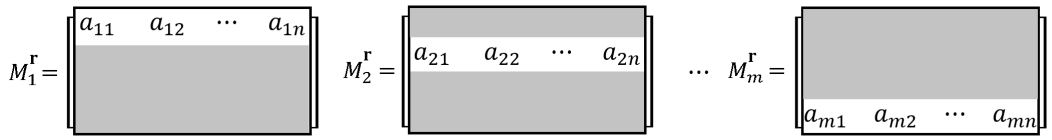


Figure 1.1 : Demonstration of a row-wise moving matrix, $M_i^r = A[i : i+a-1, :]$ for $a = 1$.

Similarly, the column-wise moving matrix $M^c \subset A$ is defined as:

$$M_j^c = A[:, j : j+a-1],\ j = 1,2,3...,n-a+1,\ M^c \in \mathbb{R}^{m \times a}$$

where a denotes the number of column-wise sub-interval. The colon ":" stands for all the elements in the row of a matrix.

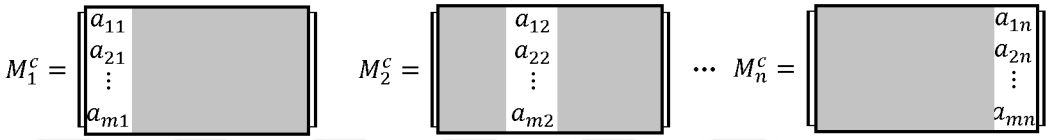


Figure 1.2 : Demonstration of a column-wise moving matrix, $M_i^c = A[i : i+a-1, :]$ for $a = 1$.

The next subsection provides a glance at concept of artificial intelligence, its types and fundamental components.

1.2 Artificial Intelligence

The development of artificial intelligence (AI) followed that of game theory [68], and early contributions to both subjects were made by pioneers like Findler and Simon [69, 70]. Game theory and artificial intelligence consist of intelligent players or agents who interact with their environment and each other, who try to optimize their actions by utilizing different reasoning and learning techniques [71]. However, the most of the first fourty years of artificial intelligence were centered around the creation and examination of autonomous agents so that the necessity of game theory were omitted [72]. By the time, game theory started to used with artificial intelligence and it helped to model some deep learning based problems. The game theoretical tools are used for the design of deep learning models [73]. Recently, various real-time problems are solved by using game theory and artificial intelligence and the studies show that the application of the game theory with artificial intelligence boosts the performance of the solution approaches [71, 74, 75].

The artificial intelligence encompasses broad scientific field. The term AI stands for the advancement of computer systems that can run the tasks which are usually need human intelligence. The term includes a wide range of methods and approaches. Artificial intelligence first appeared in 1950s whenever the computer scientists ask whether the computer could be made to think. In AI, the scientists used sufficiently large datasets of explicit rules for manipulating the knowledge [76]. On the other hand, there is a subfield, machine learning (ML), of artificial intelligence which focus on the usage of specific algorithms over data analysis and learning process. In machine learning, the data and expected answer from the data are provided to the computer and let the computer find a rule based on statistical methods. In other words, the machine learning is a part of artificial intelligence and its focus centered on the improvement of algorithms and statistical approaches which helps the computers to boost their performance on a certain task by learning from the data and expected answer without programmed in details on the contrary to the classical programming [76]. Machine learning approach needs large dataset for training, validation and testing the models. In addition to machine learning, the use of an advanced part of machine learning algorithms is popular and this subfield is known as deep learning. Deep learning (DL) is a machine learning approach which includes successive layers of representations, in other words, layers of neural networks. In other words, the deep learning models utilizes multiple layers in an artificial neural network for finding specific characteristics from the unprocessed data [73].

All the above techniques can be used wide range of tasks. However, ML includes prediction, recognition and decision-making depending on the dataset. DL is applied to more complex problems such as image and speech recognition. In general, the algorithm architecture of the approaches varies. For example, AI involves symbolic reasoning, search algorithm and systems based on rules. However, ML uses linear regression, decision trees and so on, changing according to the task. DL uses artificial neural networks with layers. From the broader perspective, artificial intelligence is defined as designing intelligent systems that runs tasks usually require human intelligence. Machine learning is a subfield of AI and defined as creating algorithms which learn from given data and decide using the observed patterns. It sometimes needs a human to intervent and correct whenever the decision is incorrect. On the

other hand, deep learning utilizes artificial neural networks to get proper conclusions without requiring human involvement.

1.2.1 Neural networks

Artificial neural network defines a computation model consisting of numerous nodes connected to each other, where each node (neurons) has a specific output function, called activation or transformation function. A neuron is a process unit representing different objects and has three types which are input, hidden and output units. An input unit gathers the signals from the real world, output unit presents the result processed by the system and lastly the hidden layer, that cannot be observed directly, is placed between input and output units [77]. A basic process unit of an artificial neural network, a neuron, is depicted in the following figure [78],

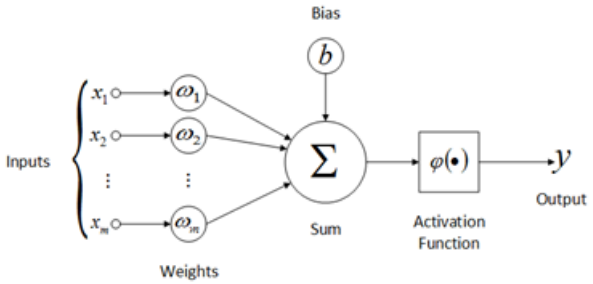


Figure 1.3 : General structure of a neuron

The neuron runs the following equation for calculations,

$$y = \phi\left(\sum_i w_i x_i + b\right) \tag{1.5}$$

where w_i are weights, b is the bias, which aims to increase or decrease the input of the activation function, ϕ is the activation function, which roughly determines the importance of the feature whether it should be passed to the next layer or not [78].

In the context of artificial neural networks, weights are the essential parameters which has an important role in the ability of the network to learn and predict the results. These parameters get adjusted while the training process in action and they are adjusted to minimize the error between the true value and the predicted value. The adjustment process is usually run by a learning algorithm such as backpropagation integrated with including gradient descent [79]. The bias term serves to adjust the activation function's results towards positive or negative side [80]. As a summary, a neural network consists

of nodes, in other words the neurons. These neurons process the input data using some mathematical tools to move the result the following neurons until the last neuron. The last neuron provides a prediction. According to error between the true value and actual value, the parameters in the mathematical tools get adjusted using proper optimization techniques.

A neural network includes several components such as layers establishing the network, input data and its targets, a loss function, an optimizer. An input provided to the neural network. Then, the data is filtered through the layer and turned into the proper representations. Next, a prediction is given by the neural networks. The loss function evaluates the loss score and the weights of the data provided by the layers are updated depending on the optimization function. The next figure demonstrates the relationship between these components [76].

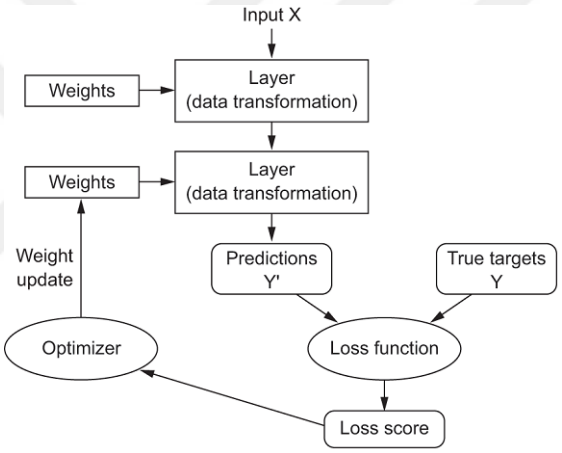


Figure 1.4 : Components of a neural network and their relationships

1.2.2 Types of neural networks

This subsection presents the most commonly used types of neural networks in artificial intelligence, which are feedforward neural networks, recurrent neural networks and convolutional neural networks.

Feedforward Neural Networks (FNN): FNNs are the first created neural networks and used widely in applications today. The information flow moves without including the loops in the FNNs so that this neural networks are called feedforward neural network. FNNs are classified as single-layered networks or multilayered networks,depending of the presence of the hidden layers. The number of the layers

are determined based on the complexity of the work. FNN are mostly used for object recognition and speech recognition systems [81, 82].

Recurrent Neural Networks (RNN): On the contrary to the FNN, RNN contains the recurrence of the operations in the form of loops and mostly used for more complex tasks such as text prediction and language generation. In RNN, the connections are allowed to move back neurons in the same layer while the connections in FNN are led from one neuron to the next one in subsequent layers without any feedback [81].

Convolutional Neural Networks (CNN): CNNs are separated from the other neural networks by its superior performance in complex tasks. In other words, CNN is a special kind of deep learning architecture, mostly known for its remarkable performance for analyzing visual imagery [73]. A CNN includes three main types of layers such as convolutional layer, pooling layer and fully connected (dense) layer. The foundational building component of a CNN is the convolutional layer, which needs several components as the input data, a filter and feature map [83]. Roughly, a convolution layer can be considered as a filter that applies to a matrix of a pixel values which are a part of a larger image [84]. The pooling layers downsample the feature maps, conducting dimension reduction, reducing the number of parameters in the input [76]. There are two type of pooling layers, which are max pooling and average pooling. In the max pooling, the filter goes through the input and choose the maximum value to send to the output array. In average pooling, it does the same task in max pooling using the average value.

1.2.3 Activation functions

Sigmoid Function [85] The sigmoid function generated an S-shaped curve between 0 and 1. The most disadvantageous part is that it is tend to the gradient vanishing problem. Gradient vanishing means that when the input value gets bigger, its derivative gets closer to zero. Therefore, it causes some difficulties for the update process during the backward propagation

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

ReLU (Rectified Linear Unit) [86] ReLU function operates by setting values at 0 as a threshold, stated as

$$f(x) = \max(0, x)$$

It simply produces 0 whenever $x < 0$ and it output the value itself whenever $x \geq 0$.

Tanh [87] Tanh function is represented as

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

It generates an S-shaped curved on [-1,1]. The larger input becomes close to 1 and smaller inputs get close to -1.

SoftMax Function [87] SoftMax function, also known as soft argmax function, is a nonlinear function and it is represented as

$$\text{softmax}(x) = \frac{e^x}{\sum_i e^{x_i}}$$

SoftMax function produces a categorical probability distribution equivalent output.

In addition to these activation functions, Leaky ReLU, Parametrized ReLU, Swish, Hexpo [88, 89] are used in neural networks depending on the problem. The following figure demonstrates some proper usage of the activation functions as is in [90].

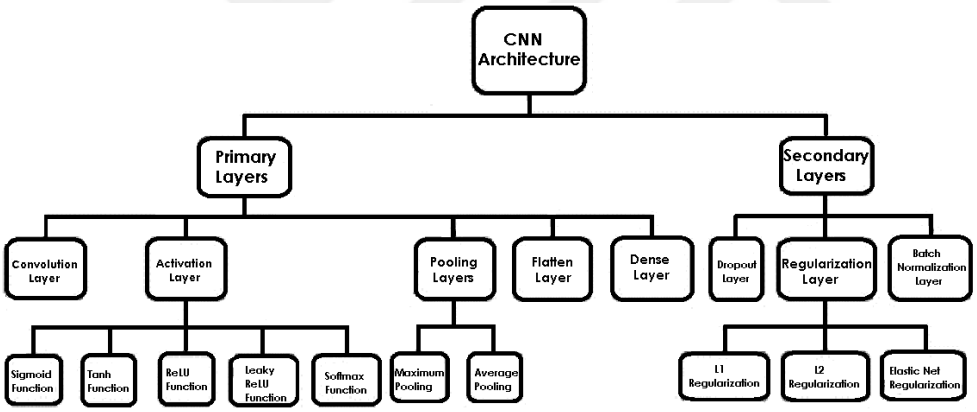


Figure 1.5 : General structure of a convolutional neural network and usage of activation functions

1.2.4 Gradient descent

Gradient is a derivative of a tensor operation, in other words, it is the concept of derivatives to functions of multidimensional inputs, is called tensors [76]. Gradient descent is one of the most often used artificial neural network optimization process [91]. Gradient descent is optimization algorithm for weights and biases terms in Equation 1.5 used in machine/deep learning. The purpose of gradient descent is to minimize an objective function using iterations. In machine/deep learning the objective function is a cost/loss function, which depends on the problem.

As an illustration, consider a simple linear regression and its cost function Mean Squared Error (MSE) [92].

$$\hat{y} = wx + b$$

where $\hat{y}$ denotes the predicted value, w and b represent the weights and the bias term, respectively. We firstly evaluate the errors between the true value y and the predicted value $\hat{y}$ as

$$e = y - \hat{y}$$

Then, we compute the MSE as

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2.$$

Next, we evaluate the gradients, in other word, the partial derivatives with respect to b and w as,

$$\frac{d\text{MSE}}{db} = -\frac{2}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i) = -\frac{2}{n}\sum_{i=1}^{n}(y_i - (wx_i + b))$$

$$\frac{d\text{MSE}}{dw} = -\frac{2}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)x_i = -\frac{2}{n}\sum_{i=1}^{n}(y_i - (wx_i + b))x_i.$$

Next, we update the parameters of gradient descent as

$$b = b - \text{lr} \times \frac{d\text{MSE}}{db}$$

$$w = w - \text{lr} \times \frac{d\text{MSE}}{dw}$$

where lr is the learning rate. This process is continued for a certain error or a total number of iterations.

Another important algorithm is stochastic gradient descent (SGD) which is an optimization algorithm in machine/deep learning to minimize the loss function. It works very similar to the gradient descent algorithm with some differences. Stochastic part of the SGD means that it uses a randomly selected subset of the dataset to evaluate the gradients in contrast to the gradient descent which uses the whole dataset for calculations. The stochastic gradient descent algorithm can be summarized as follows:

Algorithm 1 Stochastic Gradient Descent

1: Select arbitrary values for the model parameters (weights, bias), denoted as θ.
2: Let $\{x_i, y_i\}$ represent the training set for $i = 1, 2, \ldots, n$, where x_i and y_i denote the feature vector and target vector, respectively.
3: Let $J(\theta)$ represent the loss function (e.g., MSE, Cross-entropy, etc.).
4: Choose a randomly generated subset of the training dataset for each iteration.
5: Evaluate the gradients of the loss function for the selected subset.
6: Update the parameters as $\theta_{i+1} = \theta_i - \alpha \cdot \nabla J(\theta_i)$
7: Repeat until a certain stopping condition is met, such as a maximum number of iterations or convergence of the parameters.

where θ_i are the parameters in the i-th iteration, α denotes the learning rate, and $\nabla J(\theta_i)$ represents the gradients for θ_i.

1.2.5 Forward and backward propagations

Forward propagation defines the motion of the input from the first layer to the last layers. Forward propagation process starts with the input data and pass the input data through each layer using the activation functions and deliver it to the last layer.

$$z_i^{[L]} = \sum_j w_{ij}^{[L]} \cdot a_j^{[L-1]} + b_i^{[L]}$$

$$a_i^{[L]} = g(z_i^{[L]})$$

where w_{ij} weights, b_i bias and $g()$ activation function.

Backward propagation is another optimization algorithm based on the gradient descent. The algorithm begins with the outcome of the neural network and computes the error in each layer of the neural network backwardly, so that gradients are obtained for each weight and biases and these values are updated.

$$\frac{dJ}{dz^{[L]}} = \frac{dJ}{da^{[L]}} \cdot \frac{da^{[L]}}{dz^{[L]}}$$

$$\frac{dJ}{dw^{[L]}} = \frac{dJ}{dz^{[L]}} \cdot \frac{dz^{[L]}}{dw^{[L]}}$$

$$\frac{dJ}{db^{[L]}} = \frac{dJ}{dz^{[L]}} \cdot \frac{dz^{[L]}}{db^{[L]}}$$

where J cost function, $z^{[L]}$ input of activation function, $w^{[L]}$ weights, $b^{[L]}$ bias.

1.2.6 Artificial intelligence and game theory

By the mid of 90s, the computer scientist began to pay attention to the game theory due to two main reason. First reason was the challenges of the applications of the systems with computational complexities for economic problems so that the computer scientist involved in the area. The second reason was the increment in the use of the internet. Therefore, the scientists started to investigate the problems including intelligent agents [72]. Recently, the game theory and artificial intelligence approaches are used together. Tennenholtz [71] emphasized that the artificial intelligence surpasses human-level intelligence in the large-scale perfect information games. Tuyls and Nowe investigated the usage of evolutionary game theory with reinforcement learning for multi-agent systems [93]. Schuster and Yamaguchi proposed an new method with the help of game theoretical tool for modeling individual neurons [94]. Agarwal and Jaiswal presented a machine learning algorithm for iterated prisoner dilemma game and show that the artificial intelligence helps to improve the decision of the agents [95]. Madani and Hooshyar used game theory integrated reinforcement learning methods to develop optimal policies for multi-operator reservoir systems [96]. Xiao et al. examined the jamming games underwater and the authors present a power control strategy obtain by Q-Learning [97]. Rekha et al. studied how the game theory takes place in some of machine learning problems [98]. Stier et al. used Shapley value to eliminate the useless elements of a neural network [99]. Moran and Pollack presented a deep learning model that is also application of evolutionary game theory on the behaviour of the networks [100]. Albaba and Yıldız used reinforcement learning and behavioral game theory for the prediction of the highway driver under different scenarios [101]. Wu and Lisser proposed an dynamical neural network model for determining the saddle point of the stochastic zero sum games [102]. Khabbouchi et al. presented game theoretical model using machine learning for advanced wind energy management protocol [103].

The number of studies about the game theory and artificial intelligence techniques used together can be increased. These studies show that the both field has a close relationship. While game theory sometimes accelerates and improves artificial intelligence methods, sometimes artificial intelligence-supported game theory models

produce more realistic results for real-time problems. In the following section, we present an machine learning approach supporting a solution method for matrix games, which is known as the extended matrix norm method.

2. EXTENDED MATRIX NORM METHOD AND CONVERGENCE RESULTS

In this chapter, we extend and apply the matrix norm (MN) method to the nonzero-sum bimatrix games. We present preliminary results regarding the convergence of the matrix norm methods. We introduce a notation that allows us to express nonzero-sum bimatrix games as two separate matrix games, highlighting the separation concept. Then, we prove theorems regarding boundaries of the game value depending on only norms of the payoff matrix for each player of the nonzero-sum bimatrix game. Additionally, we refine the boundaries for both zero-sum and nonzero-sum matrix games, leading to an improved interval for the game value, a significant advancement for both types of games. Therefore, we succeed to find an improved interval for the game value, which is a crucial improvement for both nonzero and zero-sum matrix games. As a consequence, we can solve a nonzero-sum bimatrix game for each player approximately without solving any equations. Moreover, we modify the inequalities for the extrema of the strategy set for the nonzero-sum bimatrix games. Furthermore, we adapt the min-max theorem of the MN method for the nonzero-sum bimatrix games. Finally, we consider various bimatrix game examples from the literature, including the famous battle of sexes, to demonstrate the consistency of our approaches. We comprehensively demonstrate the converge of the matrix norm methods numerically. We also show the application of bimatrix games on a real-life problem using Covid-19 pandemic as a case and utilize repeated bimatrix games for investigating the problem.

In the literature, nonzero-sum bimatrix games are typically solved through the application of theorems or by decomposing the game into two separate matrix games [53, 104]. This possibility of separation leads researchers to utilize linear programming and other optimization methods to obtain solutions. While existing methodologies are effective for small-scale matrix games, they may encounter limitations when applied to larger or more complex scenarios. However, the solution procedure gets more tedious for obtaining a solution for large-scale matrix games due to the increasing number of inequalities and the computational cost. Thus, İzgi and Özkaya focused

on approximatelty solving and generating zero-sum matrix games by using norms of the payoff matrix [1] . In particular, İzgi and Özkaya introduced the MN method that approximately solves the zero-sum matrix game without the need of solving any equations. According to the literature, İzgi and Özkaya brought the game theory and the matrix norms together for the first time with their paper [1]. They additionally use this approach for the demonstration of the fairness of a matrix game in [105]. Further, Özkaya investigated the matrix norm method and the use of matrix norm in the game theory in detail [57]. In addition, İzgi and Özkaya apply the matrix norm method to show the necessity of agricultural insurance and show the usefulness of the method [106].

In this chapter, we are the first to extend and apply the MN method from [1] to nonzero-sum bimatrix games. Our primary focus is on the solution process of the MN method and its adaptation for bimatrix games. By implementing this approach, we aim to enhance the understanding and resolution of these complex games within the existing literature. To apply the matrix norm method to the bimatrix case, we draw on the concept from [53, 104] that allows us to decompose a bimatrix game into two separate matrix games. We introduce specific notations to facilitate this separation, enabling us to analyze each component individually. This approach lays the groundwork for effectively applying the MN method to bimatrix games. Then, we first theoretically adapt and prove theorems that provide the boundaries for the game value of corresponding players by using the same proof approach used for the matrix games in [1] with the new notation of the bimatrix games. We also demonstrate the boundaries for the extrema of the strategy set of the players for the nonzero-sum bimatrix games. Moreover, we modify the min-max theorem in [1] for the nonzero-sum bimatrix games. Additionally, we advance the boundaries for the game value in [1]. Hence, we succeed to find an improved interval, which is a crucial improvement for both nonzero and zero-sum matrix games, for the game value. In addition to these, we consider the convergence of the MN methods, and present some basic results regarding the convergence of the matrix norm approaches.

The chapter is organized as follows: In Section 2.1, we introduce a new notation for nonzero-sum bimatrix games and derive generalized theorems on game value boundaries, including inequalities for the upper and lower bounds of the strategy set.

We also present the min-max theorem and new theorems for refining game value boundaries for both zero-sum and nonzero-sum games in Section 2.2. In Section 2.3, we state and prove a theorem on the convergence of the matrix norm method, along with a corollary ensuring the existence of a solution. In Section 2.4 presents the implementation of the extended matrix norm method to bimatrix games and application of repeated bimatrix games on Covid-19 pandemic.

2.1 The Modification of the Matrix Norm Method

In this section, we extend the matrix norm method proposed in [1, 57] to nonzero-sum bimatrix games. We begin by introducing a new notation tailored for these games to facilitate the adaptation of theorems and proofs from [1, 57]. We present and thoroughly prove theorems that establish inequalities related to the game value of nonzero-sum bimatrix games. Additionally, we introduce a theorem regarding p^k_{max} and p^k_{min} representing the greatest and smallest components of each player's mixed strategy set, respectively. We derive new boundaries for the game value and adapt the min-max theorem from zero-sum matrix games to illustrate the relationship between p^k_{max} and p^k_{min} in the context of nonzero-sum bimatrix games.

Consider the real valued $A \in m \times n$ bimatrix nonzero-sum game given in Definition 1.1.3,

$$A = \begin{bmatrix} (a_{11}, b_{11}) & . & (a_{1n}, b_{1n}) \\ . & . & . \\ (a_{n1}, b_{n1}) & . & (a_{mn}, b_{mn}) \end{bmatrix}$$

where the pair (a_{ij}, b_{ij}) denotes the payoffs for Player I and Player II, respectively. We denote a_{ij} as $A[(i,j),1]$ and b_{ij} as $A[(i,j),2]$. We can represent this bimatrix as two distinct matrices, expressed as follows:

$$A^1 = \begin{bmatrix} a_{11} & . & a_{1n} \\ . & . & . \\ a_{n1} & . & a_{mn} \end{bmatrix}, A^2 = \begin{bmatrix} b_{11} & . & b_{1n} \\ . & . & . \\ b_{n1} & . & b_{mn} \end{bmatrix}$$

where $A^1 = A[(i,j),1]$ and $A^2 = A[(i,j),2]$ are the payoff matrices for the first and second players, respectively. Contrary to the zero-sum matrix game, we have two different game values in this case as v^1 and v^2 whose total sum is not a constant, that is, $v^1 + v^2 \neq c$ where c is constant as defined in Definition 1.1.2. This means that knowing the game value of any player does not give any idea about the other's game value. It is

the main reason these types of games are called nonzero-sum games [104]. Therefore, we need to solve two different matrix games. On the other hand, if $A^1 = -A^2$, then the game is a zero-sum matrix game in the common sense [53]. In this situation, one may approximately solve the game directly by using the methodology presented in [1,57]. Next definition expresses the solution of the bimatrix games using the notation presented previously in terms of linear programming approach.

Definition 2.1.1 *[42] Let A be real valued $m \times n$ bimatrix of the nonzero-sum game and v^k be the game value of the k^{th} player for $k = 1,2$. Then, the game value is obtained by $v^k = \sum_{i=1}^{m} A[(i,j),k]p_i^k$, for any j and fixed k, where p_i^k's are the elements of the mixed strategy set of the corresponding player and $\sum_{i=1}^{m} p_i^k = 1$.*

We start developing the extended matrix norm method by proving the next lemma for bimatrix games.

Lemma 2.1.2 *[42] Let A be real valued $m \times n$ bimatrix of the nonzero-sum game as in the following form*

$$A = \begin{bmatrix} (a_{11}, b_{11}) & . & (a_{1n}, b_{1n}) \\ . & . & . \\ (a_{m1}, b_{m1}) & . & (a_{mn}, b_{mn}) \end{bmatrix}$$

and the game value of the k^{th} player is v^k where $k = 1,2$. Then,

$$\text{if } v^k \geq 0, \text{ then } \frac{z}{||A^k||_\infty} \leq v^k \leq ||A^k||_1$$

and

$$\text{if } v^k \leq 0, \text{ then } -||A^k||_1 \leq v^k \leq \frac{z}{||A^k||_\infty}$$

hold where $z = v||B^k||_\infty$ for fixed k and B^k is the row-wise induced matrix, stated in Definition 1.1.8 in Chapter 1, of the bimatrix A for the related player.

Proof. We first split the bimatrix game into two different matrix games, as it is presented in [53, 104], with our notation above, say A^1 and A^2. Then, we can prove the lemma directly by following the same motivation that is used for the zero-sum matrix games in [1].

In Lemma 2.1.2, the boundaries strictly depend on the game value. In the following corollary, we eleminate the game value in the boundaries.

Corollary 2.1.2.1 *[42] Let A be real valued $m \times n$ bimatrix of the nonzero-sum game, v^k is the game value of the related player and B^k that is the row-wise induced matrix of the bimatrix A for the corresponding player. Then,*

$$\text{if } v^k \geq 1 \text{ then } \frac{||B^k||_\infty}{||A^k||_\infty} \leq v^k \leq ||A^k||_1$$

and

$$\text{if } v^k \leq -1 \text{ then } -||A^k||_1 \leq v^k \leq -\frac{||B^k||_\infty}{||A^k||_\infty},$$

hold where $k = 1,2$ and B is the row-wise induced matrix of A.

Next theorem presents the boundaries without the game value. Hence, the boundaries of the game value for a bimatrix games can be obtained using only the matrix norms of the payoff matrix for the corresponding player.

Theorem 2.1.3 *[42] Assume that A be real valued $m \times n$ bimatrix of the nonzero-sum game as in the following form*

$$A = \begin{bmatrix} (a_{11}, b_{11}) & . & (a_{1n}, b_{1n}) \\ . & . & . \\ (a_{m1}, b_{m1}) & . & (a_{mn}, b_{mn}) \end{bmatrix}$$

and the game value of the k^{th} player is v^k. Then,

$$\text{if } |v^k| \geq 1 \text{ then } \frac{||B^k||_\infty}{||A^k||_\infty} \leq |v^k| \leq ||A^k||_1$$

and

$$\text{if } |v^k| \leq 1 \text{ and } v^k \neq 0 \text{ then } \frac{1}{||A^k||_1} \leq |v^k| \leq \frac{||A^k||_\infty}{||B^k||_\infty}$$

hold where for $k = 1,2$, and B^k is the row-wise induced matrix of A^k.

Proof. We know $\frac{||B^k||_\infty}{||A^k||_\infty} \leq 1$ where $||A^k||_\infty = \max_i \sum_{j=1}^{n} |A[(i,j),k]|$ and B^k is the row-wise induced matrix of A^k, for any k. We also have $v^k \leq ||A^k||_1$ when v^k is positive and $-||A^k||_1 \leq v^k$ when v^k is negative by Lemma 2.1.2.

(i) When v^k is positive,

(a) If $v^k \geq 1$, then we have $\frac{||B^k||_\infty}{||A^k||_\infty} \leq v^k \leq ||A^k||_1$ by Corollary 2.1.2.1

(b) If $0 \leq v^k \leq 1$, then we suppose $t^{-1} = v^k$. Thus, we have, $t = \frac{1}{v^k} \geq 1$. Then, it follows $\frac{1}{||A^k||_1} \leq v^k \leq \frac{||A^k||_\infty}{||B^k||_\infty}$ by using the result in (a).

(ii) When v^k is negative,

(a) If $v^k \leq -1$, then $v^k \leq -1 \leq -\frac{||B^k||_\infty}{||A^k||_\infty}$. We reach $-||A^k||_1 \leq v^k \leq -\frac{||B^k||_\infty}{||A^k||_\infty}$ by using Corollary 2.1.2.1.

(b) If $-1 \leq v^k \leq 0$ then we assume $s^{-1} = v^k$. Hence, $s = \frac{1}{v^k} \leq -1$. We get $-||A^k||_1 \leq s \leq -\frac{||B^k||_\infty}{||A^k||_\infty}$ by the result of (a). Then we can rewrite it as, $-\frac{||A^k||_\infty}{||B^k||_\infty} \leq v^k \leq -\frac{1}{||A^k||_1}$

After the necessary arrangements for the inequalities in (i) and (ii) the result follows.■

The following proposition explains that the proposed boundaries can be applied to any bimatrix games.

Proposition 2.1.4 *[42] Any bimatrix game can be perturbed while the game value v^k changes as much as the perturbation and the strategy sets remain the same.*

Proof. One may split the bimatrix game into two different matrix games. As a result of this, we can prove it similar to Proposition 2.7 of [1] by using the new notations.

Corollary 2.1.4.1 *[42] Any bimatrix game could be reduced to a game with game value $v^k = 0$ with the same strategy set.*

Note that we can use the above corollary in order to create a nonzero-sum bimatrix game whose game value is zero for any player.

We continue to present a theorem providing lower and upper boundary for the maximum and the minimum elements of the strategy set of a bimatrix games.

Theorem 2.1.5 *[42] Let A be real valued $m \times n$ bimatrix, where $a_{ij} > 0$ for all $1 \leq i \leq m$ and $1 \leq j \leq n$, of a nonzero-sum game. Then the boundaries for the greatest and the smallest elements, p^k_{max} and p^k_{min} respectively, of the strategy set of the corresponding players are as follows*

$$p^k_{\max} \geq L^k \ \ where \ \ L^k = \max\left\{\frac{1-\frac{v^k}{||A^k||_1}}{m-1}, \frac{v^k}{||B^k||_1}\right\},$$

$$p^k_{min} \leq U^k \ \ where \ \ U^k = \min\left\{\frac{1-\frac{v^k}{||B^k||_1}}{m-1}, \frac{v^k}{||A^k||_1}\right\}$$

where B_k is the column wise induced matrix of A^k and $k = 1,2$.

Proof. The proof can be done as it is proved for Theorem 3.3 in [1].

The next theorem shows the connection between the extremum values of the strategy set. The theorem also has an essential role in the game creation process introduced in [1].

Theorem 2.1.6 ***(Min-Max Theorem) [42]*** *Let A be real valued $m \times n$ bimatrix of a nonzero-sum game. Then, $\frac{1-p_{min}^k}{m-1} \leq p_{max}^k \leq 1-(m-1)p_{min}^k$ and $1-(m-1)p_{max}^k \leq p_{min}^k \leq \frac{1-p_{max}^k}{m-1}$ hold where $k = 1,2$.*

Proof. The proof is very similar to that of Theorem 3.2, which is the min-max theorem, in [1] and hence the details are skipped here.

2.2 Extended Matrix Norm Approach: Boundaries Refinement for Game Value

The following theorems improve the boundaries for the game value of zero-sum matrix games obtained by the MN method in [1]. We present new boundaries that provides better and extended interval for the game value by defining new matrix norms.

Theorem 2.2.1 *[42] Let A be a $m \times n$ payoff matrix and v be the game value for a two person zero sum game, and define $||A||_{\bar{1}} = \min_j \sum_i |a_{ij}|$, $||A||_{\bar{\infty}} = \min_i \sum_j |a_{ij}|$. Then,*

$$\text{if } |v| \geq 1, \text{ then } L_v \leq |v| \leq U_v$$

and

$$\text{if } |v| \leq 1, \text{ and } |v| \neq 0, \text{ then } (U_v)^{-1} \leq |v| \leq (L_v)^{-1}$$

while $L_v = \max\left\{\frac{||B||_\infty}{||A||_\infty}, \frac{||B||_1}{||A||_1}\right\}$ and $U_v = \min\left\{||A||_{\bar{1}}, ||A||_{\bar{\infty}}\right\}$, and B is the row-wise induced matrix of A.

Proof. For $|v| \geq 1$: The demonstration for the left-hand side (L_v) of the inequality can be done directly using the fact that of a zero-sum game by taking into account the Theorem 2.5 in [1] in the view of Player I and Player II, separately. Similarly, one can show that the right side of this inequality holds by considering the basic probability assumption that is the strategy set elements are less than or equal to 1 (i.e. $p_i \leq 1$ for all $1 \leq i \leq n$) since $v = \sum_{i=1}^{m} p_i a_{ij}$. Correspondingly, the proof can be done by following up the same steps above for $|v| \leq 1$, as well. ■

Following theorem improves the boundaries obtained by Theorem 2.2.1 using the p_{min} and p_{max} value estimated by Theorem 2.1.5.

Theorem 2.2.2 *[42] Let A be real valued $m \times n$ payoff matrix, where $a_{ij} > 0$ for all $1 \leq i \leq m$ and $1 \leq j \leq n$, of a two person zero sum game. Then, $p_{min}||A||_1 \leq v \leq p_{max}||A||_{\bar{1}}$ holds while $||A||_{\bar{1}} = \min_j \sum_i |a_{ij}|$, and p_{min} and p_{max} are the smallest and greatest components in the mixed strategy set, respectively.*

Proof. Let $||A||_{\bar{1}} = \sum_{i=1}^{m} |a_{ij}|$ for fixed $1 \leq j_0 \leq n$. We know $v = \sum_{i=1}^{m} p_i a_{ij}$ for any $j = 1, ..., n$ and $p_i \in [0,1]$. We can also write $v = \sum_{i=1}^{m} p_i a_{ij} = \sum_{i=1}^{m} p_i |a_{ij_0}| \leq \sum_{i=1}^{m} p_{max} |a_{ij_0}|$ by using these facts. Then, the result follows. The rest of the proof can be done similarly. ■

It is clear that Theorem 2.2.2 can be also applied to nonzero-sum bimatrix games using the similarity with the zero-sum matrix games while solving the problem.

Remark 2.2.3 *[42] In order to solve a bimatrix game, after splitting the bimatrix game into two different matrix games, it is crucial to realize that the v_{app} should stay in the optimized interval obtained from Theorem 2.2.1 and Theorem 2.2.2. Moreover, the strategy set's elements must be selected in the consideration of Theorem 2.1.5 and principles of the probability and game theories. Then, one may distribute the strategies in any order by obeying these rules.*

Remark 2.2.4 *[42] The game creation process for bimatrix games can be completed by using the methodology, which is described in [1], for zero-sum matrix games.*

Finally, we summarize our extended approach for the solution of bimatrix games in Figure 2.1 which might be a practical tool for the players.

Figure 2.1 : Solution steps' diagram for the two-person nonzero-sum bimatrix games

2.3 Convergence Results of the MN Approaches

The convergence of the iterative numerical methods is essential. Therefore, in this section, we aim to discuss some theoretical preliminary results about the convergence of the MN method (and also the EMN method).

Theorem 2.3.1 *[42] Let v_{app} be approximated game value obtained by MN approaches while $P_{est}^{j} \in R^{1\times m}, j = 1,2,...,m!$ presents the all possible strategy sets of zero sum matrix game with positive entries $A \in \mathbb{R}^{m\times n}$ payoff matrix such that $P_{est}^{j} a_{ij_0} = v_{app}$, for all $1 \leq i \leq m$ and $1 \leq j_0 \leq n$. Then*

$$\lim_{||P||\to 0} v_{app} = v$$

while v is the exact game value of the matrix game with $P \in \mathbb{R}^{1\times m}$ strategy set and, $||P|| = \max\limits_{1\leq j\leq m!} \max\limits_{1\leq i\leq m} |P_{est}^{j}(i) - P(i)|$.

Proof. Let p_{max} and p_{min} be determined with respect to the Theorem 2.1.5 and Theorem 2.1.6 for $k = 1$ (or Theorem 1.1.11 and Theorem 1.1.12 in Section 1). Then, we can determine the rest of the strategy set by obeying Remark 2.2.3 and, assume that $P_{est} = \{p_{min}, p_{max}, p_1, p_2, ..., p_{m-2}\} \in \mathbb{R}^{1\times m}$ represents one of the possible strategy set. Therefore we have $m!$ different orders of the elements of the strategy set P_{est}. This implies that we have $m!$ alternatives for the strategy set which can be considered through the approaches $(i.e.\ P_{est}^{1} = \{p_{min}, p_{max}, p_1, p_2, ..., p_{m-2}\}, P_{est}^{2} = \{p_{max}, p_{min}, p_1, p_2, ..., p_{m-2}\}, ..., P_{est}^{m!} = \{p_{m-2}, ..., p_2, p_1, p_{max}, p_{min}\})$. In short, we show that all the possible alternatives of

the strategy set by $P^j_{est} = \{p^j_{min}, p^j_{max}, p^j_1, p^j_2, ..., p^j_{m-2}\}$ while $j = 1,2,3,...,m!$. Now, let us show the estimation errors with $e^j = v^j_{app} - v$, then

$$\begin{aligned} e^j &= v^j_{app} - v \\ &= \left[p^j_1 a_{1j} + p^j_2 a_{2j} + ... + p^j_{(m-1)} a_{(m-2)j} + p^j_{min} a_{(m-1)j} + p^j_{max} a_{mj}\right] \\ &- \left[p_1 a_{1j} + p_2 a_{2j} + ... + p_{(m-2)} a_{(m-2)j} + p_{m-1} a_{(m-1)j} + p_m a_{mj}\right] \\ &= \Big[(p^j_1 - p_1)a_{1j} + (p^j_2 - p_2)a_{2j} + ... + (p^j_{(m-1)} - p_{m-2})a_{(m-2)j} \\ &+ (p^j_{min} - p_{m-1})a_{(m-1)j} + (p^j_{max} - p_m)a_{mj}\Big] \end{aligned}$$

can be obtained for any one of the possible strategy set P^j_{est} obtained by MN approaches where $P = \{p_1, p_2, ..., p_m\}$ is the exact strategy set of matrix game A. Thus,

$$\begin{aligned} \lim_{||P||\to 0} |v^j_{app} - v| &= \lim_{||P||\to 0} \Big|(p^j_1 - p_1)a_{1j} + (p^j_2 - p_2)a_{2j} + ... + (p^j_{(m-1)} - p_{m-2})a_{(m-2)j} \\ &+ (p^j_{min} - p_{m-1})a_{(m-1)j} + (p^j_{max} - p_m)a_{mj}\Big| \\ &\leq a_{1j}\cdot 0 + a_{2j}\cdot 0 + ... + a_{(m-2)j}\cdot 0 + a_{(m-1)j}\cdot 0 + a_{mj}\cdot 0 \\ &= 0 \end{aligned}$$

by Sandwich theorem since $0 \leq |p^j_i - p_i| \leq ||P||$ for all $i = 1,2,...m$ and $j = 1,2,...,m!$ where $||P|| = \max_{1\leq j\leq m!} \max_{1\leq i\leq m} |P^j_{est}(i) - P(i)|$ ■.

Remark 2.3.2 *[42] It is clear from Theorem 2.3.1 that, by MN approaches, we can find possible strategy set P_{est} that is sufficiently close to actual strategy set P of zero-sum matrix game A if $||P|| \to 0$ is possible. On the other hand, there are some zero-sum matrix games whose maximum and minimum strategies are not close enough to each other for each $\varepsilon > 0$ such as $p_{max} - p_{min} > \varepsilon$ (for instance, see Example 4 in Section 2.4.1). This implies that the norm of P may never approach to 0 (i.e. $||P|| \geq (p_{max} - p_{min}) \not\to 0$). In these cases, the following corollary guarantees the existence of at least one strategy set obtained by MN approaches where the corresponding v_{app} converges to v.*

Corollary 2.3.2.1 *[42] Let A be real valued $m \times n$ payoff matrix, where $a_{ij} > 0$ for all $1 \leq i \leq m$ and $1 \leq j \leq n$, of a two person zero sum matrix game. Then, there exists*

at least one strategy set, $P_{est}^{j_0}$ where $1 \leq j_0 \leq m!$, obtained by MN approaches such that the corresponding v_{app} approaches to actual game value v of zero-sum matrix game A while $||P|| \to 0$ is not possible, in general.

Proof. Assume that A be a zero-sum matrix game whose maximum and minimum strategies are not close enough to each other for each $\varepsilon > 0$ such as $p_{max} - p_{min} > \varepsilon$, thus $||P|| \not\to 0$. On the other hand, the Theorem 2.1.5 and Theorem 2.1.6 guarantee that there are some p_{max}^{j} and p_{min}^{j} such that $p_{max} - p_{max}^{j_0} < \varepsilon$ and $p_{min} - p_{min}^{j_0} < \varepsilon$ for all $\varepsilon > 0$ and some $j_0 \in [1, m!]$. This implies that there exists at least one strategy set $P_{est}^{j_0}$ obtained with MN approaches where the norm of the corresponding P converges to 0 such as $||P^{j_0}|| = \max_{1 \leq i \leq m} |P_{est}^{j_0}(i) - P(i)| \to 0$. Consequently,

$$\lim_{||P^{j_0}|| \to 0} v_{app} = v$$

by Theorem 2.3.1 ■.

2.4 Applications of Extended Matrix Norm Method to Bimatrix Games

In this section, we aim to show the consistency of the extended Matrix Norm approach for the bimatrix game and application of repeated bimatrix games on Covid-19 in two different subsections. In Section 2.4.1, we aim to demonstrate the consistency of the EMN method for bimatrix games. In Section 2.4.2, we present a comprehensive application of repeated bimatrix games for the risk of infection during Covid-19 pandemic. We consider three countries, Türkiye, Italy and South Korea, to run our analysis during three different stages of pandemic which are the start, the spread and the end.

2.4.1 Illustrations for the EMN method on bimatrix games

We analyze two well-known examples from game theory: the Battle of the Sexes and another illustrative bimatrix game. For further exploration of game construction and solutions for two-person matrix games, readers can refer to [1, 57]. In the following examples, we first decompose the bimatrix game into two separate matrix games, applying both the matrix norm and extended matrix norm approaches. This allows us to explicitly illustrate the refinement effects of the extended matrix norm method, as stated in the relevant theorems. The final two examples focus on validating the

consistency of the results related to the convergence of the matrix norm method, as outlined in Theorem 2.3.1 and Corollary 2.3.2.1.

Example 1. [42] Consider the two-person nonzero-sum game with the bimatrix presented in [53],

$$A = \begin{bmatrix} (3,3) & (0,2) \\ (2,1) & (5,5) \end{bmatrix}$$

and the related payoff matrices of Player I and Player II, respectively, are

$$A_1 = \begin{bmatrix} 3 & 0 \\ 2 & 5 \end{bmatrix}, A_2 = \begin{bmatrix} 3 & 2 \\ 1 & 5 \end{bmatrix}$$

The given bimatrix game values in [53] are $(v^1, v^2) = (2.5, 2.6)$ and the mixed strategy sets are $S^1 = (0.5, 0.5)$ and $S^2 = (0.6, 0.4)$ for Player I (row player) and Player II (column player), respectively.

We now evaluate the interval for the game value, v^k, with extended MN method presented in previous section. To show the contributions, we first obtain intervals for each player by using Theorem 2.1.3 as follows:

$$\frac{3}{7} \leq v^1 \leq 5 \tag{2.1}$$

$$\frac{4}{7} \leq v^2 \leq 6 \tag{2.2}$$

Then, we use Theorem 2.2.1 to obtain the boundaries for the game values of each player as

$$1 \leq v^1 \leq 5$$

$$\frac{5}{6} \leq v^2 \leq 5$$

At this stage, in order to use Theorem 2.1.5 directly, we need to choose a dummy game value for the players from the corresponding interval as it is done in the solution that of Example 3 in [1], as an example choose them as an average of each interval as $v^1 = 3$ and $v^2 = 2.92$. Now, if we apply Theorem 2.1.5 for each player under the assumptions, then we have $p^1_{max} \geq L_1 = max\{\frac{1-\frac{3}{5}}{2-1}, \frac{3}{5}\} = 0.6$ and $p^2_{max} \geq L_2 = max\{\frac{1-\frac{2.92}{6}}{2-1}, \frac{2.92}{5}\} = 0.58$. Then, we select $p^1_{max} = 0.65$ and $p^2_{max} = 0.6$ in the light of the inequalities for p^k_{max} where $k = 1, 2$. It is clear that $p^1_{min} = 0.35$ and $p^2_{min} = 0.4$ since $p^k_{max} + p^k_{min} = 1$. Therefore, the mixed strategy sets in this scenario can be preferred as $S^1 = (0.35, 0.65)$ and $S^2 = (0.6, 0.4)$.

Moreover, we can obtain a better interval for the game values after the above evaluation by using Theorem 2.2.2 as, $0.35 \times 5 \leq v^1 \leq 0.65 \times 5$ and $0.4 \times 6 \leq v^2 \leq 0.6 \times 5$.

$$1.75 \leq v^1 \leq 3.25 \tag{2.3}$$

$$2.4 \leq v^2 \leq 3 \tag{2.4}$$

Furthermore, if we compare the intervals of the game value for each player obtained by Theorem 2.1.3 and Theorem 2.2.2, Equation (2.1) and (2.2) to Equation (2.3) and (2.4), respectively, we can see that Theorem 2.2.2 refines the boundaries of the game value. This iterated application of the MN method (Theorem 2.2.1 and Theorem 2.2.2, respectively) shows that we obtain a better interval for the game values. This refinement is one of the advantages parts of the extended MN approach for each player compared with MN approach in Chapter 1.

By considering the refinement interval for each game value, if we apply Theorem 2.1.5, we may obtain better strategy elements as $p^1_{max} \geq L_1 = max\{\frac{1-\frac{2.5}{5}}{2-1}, \frac{2.5}{5}\} = 0.5$ and $p^2_{max} \geq L_2 = max\{\frac{1-\frac{2.7}{6}}{2-1}, \frac{2.7}{5}\} = 0.55$. The refined strategy sets can be reconsidered as $S_1 = (0.45, 0.55)$ and $S_2 = (0.6, 0.4)$. Finally, in order to see the estimation errors, we calculate v^k_{app} for the players, $v^1_{app} = (3 \times 0.45) + (2 \times 0.55) = 2.45$ and $v^2_{app} = (3 \times 0.6) + (1 \times 0.4) = 2.6$. Hence, the errors in the approximated game values are $e^1 = |2.5 - 2.45| = 0.05$ and $e^2 = |2.6 - 2.6| = 0$. One may create other scenarios with a different selection of a dummy game value and strategies which might result in smaller errors in the approximated game values.

Example 2. [42] A famous example of a bimatrix game is the battle of sexes. We have a new married couple and they want to go somewhere together at night. The man likes football (option F), but the woman wants to go theater (option T). If the couple sees the theater, the payoff for the man is 1 and it is 4 for the woman. If the couple goes to the stadium, then the payoffs are vice versa, that are, 4 for the man and 1 for the woman. If the couple joins different activities, they both are unhappy. The bimatrix of the game is

$$A = \begin{bmatrix} & T & F \\ T & (1,4) & (0,0) \\ F & (0,0) & (4,1) \end{bmatrix}$$

First, we consider the game for the man's side. Therefore, the respective payoff matrices for the man and woman are as follows

$$A_1 = \begin{bmatrix} & T & F \\ T & 1 & 0 \\ F & 0 & 4 \end{bmatrix}, A_2 = \begin{bmatrix} & T & F \\ T & 4 & 0 \\ F & 0 & 1 \end{bmatrix}$$

This game may be considered as cooperative game. The strategy set $S^1 = \{0.8, 0.2\}$ and the game value $v^1 = 0.8$ are given in [104]. Now, we use MN approach to get estimations. By using the Theorem 2.1.3, we have $\frac{1}{0+4} \leq v^1 \leq \frac{0+4}{1+0}$. Simply,

$$0.25 \leq v^1 \leq 4 \tag{2.5}$$

Now, if we apply extended MN method directly using Theorem 2.2.1, we will have the following:

$$0.25 \leq v^1 \leq 1$$

In order to find the lower/upper boundaries for the minimum-maximum elements of the strategies, we use Theorem 2.1.5 for $m = 2$, $p^1_{max} \geq L^1$ where $L^1 = max\{1 - \frac{0.8}{0+4}, \frac{0.8}{1}\} = 0.8$. Similarly, we apply the same theorem for $p^1_{min} \leq U_1$ where $U^1 = min\{1 - \frac{0.8}{1+0}, \frac{0.8}{4}\} = 0.2$. Hence, we get $p^1_{max} \geq 0.8$ and $p^1_{min} \leq 0.2$.

Moreover, if we use Theorem 2.2.2 for $p^1_{max} = 0.81$ and $p^1_{min} = 0.19$, we have $(0.19)4 \leq v_{app} \leq (0.81)1$. Therefore, according to Remark 2.2.3, the optimized boundaries for the game value is:

$$0.76 \leq v^1 \leq 0.81 \tag{2.6}$$

Thus, the results in Equation (2.5) and (2.6) contain the actual game value. Furthermore, the refinement for the boundary of the game value can be seen explicitly from the comparison of Equation (2.5) and (2.6). One can obtain a similar interval for the woman's situation by using the payoff matrix A_2, as well.

Example 3. [42] Let A be a payoff matrix of Rock-Paper-Scissors game (RPS) with the strategy set $P = \{0.33, 0.33, 0.33\}$ and the game value is $v = 0$ [105]:

$$A = \begin{bmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{bmatrix}$$

It is obvious that $p_{max} = p_{min} = 0.33$ that are maximum and minimum strategies of RPS game are sufficiently close to each other. Therefore, this is one of the good example to show the convergence of MN approaches, while $||P|| \to 0$ is possible, under the light

of Theorem 2.3.1. directly. First, we need to perturbate the payoff matrix A of RPS game by Proposition 1.1.10 in Chapter 1 in order to get rid of the negative entries and reduce it into the applicable form of the Matrix Norm approaches. For this purpose, we select any $x = 3$ that makes all entries positive and ease our calculations since the strategy set is preserved and the game value is only perturbed 3 units [1]. Then, we obtain the following perturbed payoff matrix with the same strategy set P above while the perturbed game value is $v_p = 0 + 3 = 3$,

$$A_p = \begin{bmatrix} 3 & 2 & 4 \\ 4 & 3 & 2 \\ 2 & 4 & 3 \end{bmatrix}$$

Then, we calculate the corresponding matrix norms to use the theorems as $||A_p||_\infty = 9$, $||B_p||_\infty = 9$, $||A_p||_1 = 9$ and $||B_p||_1 = 9$. Next, we use Theorem 2.1.3 and obtain the boundaries for the game value, $1 \leq v_p \leq 9$ and choose $v_{dummy} = 5$ as an example. Later, we evaluate upper and lower bounds for p_{min} and p_{max} by Theorem 2.1.5, respectively, find the following boundaries: $p_{min} \leq U, U = min\{0.22, 0.56\}$, and select $p_{min} = 0.21$ and $p_{max} \geq L, L = max\{0.22, 0.56\}$, and choose $p_{max} = 0.57$. Finally, we obtain all possible strategy sets by obeying Remark 2.2.3 as follows:

$$P^1 = \{0.22, 0.21, 0.57\} \text{ and } v_{app1} = 2.64$$
$$P^2 = \{0.22, 0.57, 0.21\} \text{ and } v_{app2} = 2.65$$
$$P^3 = \{0.21, 0.57, 0.22\} \text{ and } v_{app3} = 2.64$$
$$P^4 = \{0.21, 0.22, 0.57\} \text{ and } v_{app4} = 2.65$$
$$P^5 = \{0.57, 0.22, 0.21\} \text{ and } v_{app5} = 2.64$$
$$P^6 = \{0.57, 0.21, 0.22\} \text{ and } v_{app6} = 2.65$$

Under the above scenarios, it is clear that $||P|| = 0.24$ is not sufficiently close to zero for all possible strategy sets, therefore most of the approximated game values are not close enough to $v_p = 3$, and the mean error is 0.355. To obtain better convergence results by MN approaches and present the importance of the Theorem 2.2.2 which is one of the most important contribution, we refine the boundaries of the game value by Theorem 2.2.2 and obtain the new interval for the game value as $1.89 \leq v_p \leq 5.13$. Then, we again determine a dummy game value in this refinement interval as an example $v_{dummy} = 3.51$. After that, we update p_{min} and p_{max} by Theorem 2.1.5 as $p_{min} \leq U, U =$

$min\{0.305, 0.39\}$, select $p_{min} = 0.3$ and $p_{max} \geq L, L = max\{0.305, 0.39\}$, and choose $p_{max} = 0.39$. Finally, we recreate the possible strategy sets by obeying Remark 2.2.3 as follows:

$$P^1 = \{0.39, 0.30, 0.31\} \text{ and } v_{app1} = 2.92$$
$$P^2 = \{0.39, 0.31, 0.30\} \text{ and } v_{app2} = 2.91$$
$$P^3 = \{0.31, 0.39, 0.30\} \text{ and } v_{app3} = 2.91$$
$$P^4 = \{0.31, 0.30, 0.39\} \text{ and } v_{app4} = 2.92$$
$$P^5 = \{0.30, 0.31, 0.39\} \text{ and } v_{app5} = 2.92$$
$$P^6 = \{0.30, 0.39, 0.31\} \text{ and } v_{app6} = 2.91$$

In the latter scenario, if we compare the corresponding possible alternative results with the former one, we can state that while $||P||$ gets closer to 0 (i.e. it reduces from $||P|| = 0.24$ to $||P|| = 0.06$), the number of the relatively convergent strategies is almost the same while the respective approximated game value approaches better to the actual game value (i.e. $||P|| = 0.06$ and $v_{app2} - v_p = 0.09$ while the mean error reduces to 0.085). To make it more clear, we consider the different scenarios and analyze the behavior of the approximated solutions in terms of the mean error in Figure 2.2.

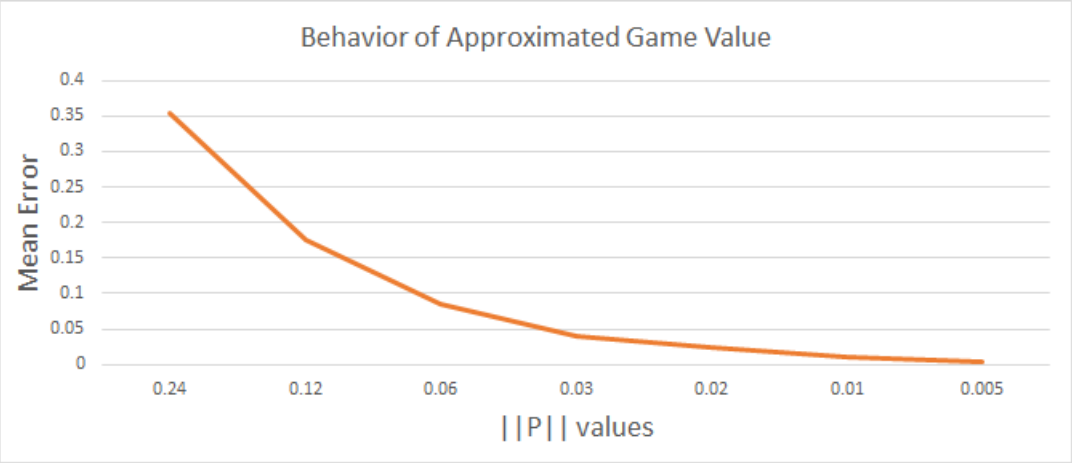


Figure 2.2 : Convergence of MN method: Mean Error vs $||P||$ values

This figure confirms the results which are promised in Theorem 2.3.1, and shows the consistency of the MN approach. Clearly, we notice that the following statement

$$\lim_{||P|| \to 0} |v_{app} - v_p| = 0$$

satisfies even though we perturb back the game value $(v_{app} - x) - (v_p - x)$ by the properties of the limit. In the first selection of $v_{pdummy} = 5$, $||P|| = 0.24$ and $v_{app} -$

$v_p = 0.36$ but the elements of the possible strategy sets are not similar to the actual mix strategy set. However, in the second selection of $v_{pdummy} = 3.51$ obtained by Theorem 2.2.2, we have $||P|| = 0.06$ and $v_{app} - v_p = 0.09$. We see that $||P||$, in this case, is sufficiently close to zero and therefore we obtain a better possible strategy set as stated in Theorem 2.3.1. If we choose v_{dummy} that results $||P|| = 0$, we can find an approximated game value exactly the same as the actual game value as it is supported by Figure 2.2.

Example 4. [42] In this example, we want to illustrate what Corallary 2.3.2.1 states with an application. Let A be real valued 3×3 payoff matrix of a game with the strategy set $P = \{0.25, 0.5, 0.25\}$ and the game value is $v = 5$ [107]:

$$A = \begin{bmatrix} 5 & 6 & 3 \\ 6 & 3 & 8 \\ 3 & 8 & 1 \end{bmatrix}$$

In order to use the theorems, we need some precalculations such as the related matrix norms: $||A||_\infty = 17$, $||B||_\infty = 14$, $||A||_1 = 17$, and $||B||_1 = 14$. Firstly, we use Theorem 2.1.3 to find an interval for the game value and choose as a dummy game value from the interval. We calculate the interval as $0.82 \leq v \leq 17$, and select $v_{dummy} = 9$, as an example. Next, we compute upper and lower bound for p_{min} and p_{max}, respectively, by using Theorem 2.1.5. Then, we obtain the upper bound for $p_{min} \leq U, U = min\{0.18, 0.53\}$, and choose $p_{min} = 0.14$ as an example since $p_{min} \leq 0.18$, . After, we calculate the lower bound for p_{max} by using the same theorem as $p_{max} \geq L, L = max\{0.24, 0.64\}$ and choose it as $p_{max} = 0.7$. Finally, we decide the element of the strategy set by considering Remark 2.2.3 and write the possible strategy sets as follows:

$$P^1 = \{0.7, 0.14, 0.16\} \text{ and } v_{app1} = 3.38$$
$$P^2 = \{0.7, 0.16, 0.14\} \text{ and } v_{app2} = 3.52$$
$$P^3 = \{0.16, 0.7, 0.14\} \text{ and } v_{app3} = 4.18$$
$$P^4 = \{0.16, 0.14, 0.7\} \text{ and } v_{app4} = 2.30$$
$$P^5 = \{0.14, 0.7, 0.16\} \text{ and } v_{app5} = 4.22$$
$$P^6 = \{0.14, 0.16, 0.7\} \text{ and } v_{app6} = 2.40$$

Even though, we have some approximated game value that are relatively close to the actual game value with some error, such as 0.82 and 0.78 belonging to the approximated game value obtained by the strategy sets P_3 and P_5, we know that a better strategy set can be found that is indicated by Theorem 2.3.1 since $||P|| = 0.45$ does not sufficiently close to zero. Therefore, we need to reconsider the maximum element in the possible strategy sets in order to find a more fitting strategy set such that satisfying $v_{app} - v = 0$.

In the light of these fact, we firstly start with refining the boundaries of the game value by using Theorem 2.2.2 in order to update v_{dummy} as following, $0.14 \times 17 < v \leq 0.7 \times 12$ and the resulting interval is,

$$2.38 \leq v \leq 8.4 \tag{2.7}$$

Then, let us select another dummy game value from inequality in (2.7) as an example $v_{dummy} = 7$ in order to reconsider the maximum element. We use Theorem 2.1.5 for the computation of upper and lower bound for p_{min} and p_{max} again, respectively. We evaluate them as $p_{min} \leq U, U = min\{0.25, 0.41\}$ and $p_{max} \geq L, L = max\{0.29, 0.5\}$. The upper and lower boundaries are $p_{min} \leq 0.25$ and $p_{max} \geq 0.5$. Later, we choose $p_{max} = 0.52$ and $p_{min} = 0.22$. Next, we create $3! = 6$ different strategy sets by the the facts stated in Remark 2.2.3 and write them as follows:

$$P^1 = \{0.22, 0.26, 0.52\} \text{ and } v_{app1} = 3.26$$
$$P^2 = \{0.22, 0.52, 0.26\} \text{ and } v_{app2} = 4.96$$
$$P^3 = \{0.26, 0.22, 0.52\} \text{ and } v_{app3} = 3.06$$
$$P^4 = \{0.26, 0.52, 0.22\} \text{ and } v_{app4} = 5.02$$
$$P^5 = \{0.52, 0.22, 0.26\} \text{ and } v_{app5} = 3.58$$
$$P^6 = \{0.52, 0.26, 0.22\} \text{ and } v_{app6} = 3.86$$

Here, we can see that $||P|| = 0.27$. Since $||P||$ is not sufficiently close to zero for all possible strategy sets, it means that we can say by Corollary 2.3.2.1 that, there exists at least one possible distribution of strategies such that almost satisfying the statement in Theorem 2.3.1 since $||P||$ is not exactly zero in this case. Thus, it is clear that $e = v_{app} - v \to 0$ as $||P|| \to 0$ for the strategy set P_4 (or P_2) and its game value is $v_{app2} = 5.02$. We can say that if we choose and distribute the elements of the

strategy set properly by keeping Remark 2.2.3 and other related theorems, we can find a strategy set such that $||P|| \to 0$, and therefore we can find the actual game value by the MN methods while $v_{app} \to v$.

2.4.2 Application of repeated bimatrix games: Case of Covid-19 pandemic

The Coronavirus disease 2019 (Covid-19) raised in Wuhan, China in 2019 and swiftly spreads to other cities in China in a couple of days. Then, it was declared as a public health emergency by World Health Organization (WHO) on 30.01.2020. Due to late precautions taken by China, the infected people carried the virus all around the country and went to other countries. The red alert is given all around the globe and all countries started to take some precautions such as stopping the international flights, closing their borders with neighbors, and declaring regional or national lockdown. The medical researchers try to understand the virus while the scientists focused on modeling the expansion by using mathematical tools [108].

This example examines the effects of general and self-quarantine measures on the spread of the first wave of the Covid-19 pandemic through a game-theoretical perspective [108]. We focus specifically on the impacts of quarantine during three distinct stages of the pandemic: the start, the spread, and the end. Our analysis involves three countries-South Korea for representing self-quarantine, and Italy and Türkiye for representing general quarantine. We introduce a formula essential for constructing payoff matrices and outline a general procedure for creating these matrices for each stage of the pandemic [108]. Using the average daily diagnosis and testing numbers, we generate payoff bimatrices for each country at each stage. Our goal is to identify the optimal strategy for managing the pandemic and to examine the necessity of continuing quarantine measures through a repeated game framework. This allows us to convert the game specifically for the spread stage into a repeated game for each country. Ultimately, we derive the Nash equilibrium for each game's respective stage. The findings indicate that quarantine significantly influences infection rates and pandemic spread across all levels. Furthermore, we compare the death rates among the countries studied, revealing a strong correlation with the quarantine requirements identified through our game-theoretical analysis [108].

We begin with stating and preferring to use the following average formula to summarize and reduce the data for each stage of the pandemic into the applicable form of the bimatrix game since we relatively determine the stages of the pandemic according to the daily data for each country [108].

$$p = \frac{1}{n} \sum_{i=1}^{n} \frac{d_i}{t_i} \tag{2.8}$$

Here,

p: The average of the probability of a person being infected.

d_i: The number of the infected person whose PCR test is positive in the i^{th} day.

t_i: The number of PCR test which is done in the i^{th} day.

n: The number of the day in the stage (i. e. the beginning, peak or slow-down) of the pandemic.

$p^{'}$: The average probability of a person not being infected is $p^{'} = 1 - p$.

Next, we generate the payoff matrix for each stage of the pandemic for each country by using the quantitie(s) obtained by Equation 2.8 for the payoff matrices in our analysis. In the lights of these information and the following assumption and notation, we create the entries of the payoff matrix.

The general procedure for creation of the payoff matrix:

1. The player strategy set consist of two elements as $S = \{Q, Q^{'}\}$ where Q : Keeping the quarantine and $Q^{'}$: Breaking the quarantine.

2. The payoff for the strategy Q is $p^{'}$ and $Q^{'}$ is $-p$, which are obtained by Equation 2.8, since $p^{'}$ is a positive option that means it is the action keeping the player safe. The breaking quarantine affects the person negatively therefore we use minus sign for the value of average probability of a person being infected to present this effect, $-p$, explicitly.

3. If the both players choose the action Q, which is (Q, Q) strategy pair, they both get the payoff $p^{'}$ since they select to prevent themselves from the infection. Equivalently, they both get $(p^{'}, p^{'})$ as their payoff.

4. If one of the players choose $Q^{'}$, in other words one selects to break the quarantine , i.e. $(Q,Q^{'})$ or $(Q^{'},Q)$ strategy pair, the player chooses $Q^{'}$ strategy increases the risk of infection so that his/her payoff decreases and the payoff belong to $Q^{'}$ is $-p$. Therefore, their payoffs are $(p^{'},-p)$ and $(-p,p^{'})$ regard to $(Q,Q^{'})$ or $(Q^{'},Q)$ strategies, respectively.

5. If the both players decide to break the quarantine, they both increase the risk of infection and their payoffs also negatively double since they both are out.

Under the favor of the above steps, we create the payoff matrix as,

$$CountryName_{Stage} = \begin{bmatrix} & Q & Q^{'} \\ Q & (p^{'},p^{'}) & (p^{'},-p) \\ Q^{'} & (-p,p^{'}) & (-2p,-2p) \end{bmatrix}$$

where Q : Keeping the quarantine and $Q^{'}$: Breaking the quarantine.

Finally, we perform game theoretical tools for the case analysis of the pandemic in South Korea, Italy and Türkiye in order to explain and understand, respective, the self or general quarantine effects during to Covid-19 pandemic in the first wave.

Case Studies [108]

Case I: South Korea

In this section, we analyze the effects of self-quarantine during the first wave of the Covid-19 pandemic in South Korea. Throughout the analysis of South Korea, we make use of the data announced from 01.02.2020 to 09.04.2020 even though there are some Covid-19 cases diagnosed before these dates which are negligible and does not affect the analysis, since the governments all around the world did not update their statistics regularly in the very beginning of the spread [109, 110].

In the first stage, we take account and analyze the data during 1-25 February since the average of the numbers of test and diagnose in daily base are small. We name this stage in the time period, 1-25 Feb., as the start.

In order to create the payoff matrix, we follow the procedure of the payoff matrix-creation given above:

Step (1). There are two strategies Q and $Q^{'}$ which is valid for all stages, stay in quarantine and break the quarantine, respectively.

Step (2). We first evaluate the average of the probability of a person being infected and the average probability of a person not being infected as are $p = 0.0160$ and $p^{'} = 1 - p = 0.9840$, respectively, by Equation 2.8.

Step (3-5). We construct the payoffs for the strategies (Q,Q), $(Q,Q^{'})$, $(Q^{'},Q)$ and $(Q^{'},Q^{'})$. Later, we generate the payoff bimatrix SK_{start} by using the payoffs p and $p^{'}$, where SK denotes South Korea. Then, we get the following payoff matrix for the first stage,

$$SK_{start} = \begin{bmatrix} & Q & Q^{'} \\ Q & (0.9840, 0.9840) & (0.9840, -0.0160) \\ Q^{'} & (-0.0160, 0.9840) & (-0.0320, -0.0320) \end{bmatrix}$$

where Q : Keeping the quarantine and $Q^{'}$: Breaking the quarantine.

In order to see the gain/loss of the players, we find Nash equilibrium point of this game as known in the literature: First we put a mark on the maximum value of each column in the first entry of the binary. Then, we do the same for the maximum value in each row of the second entry in the binary [111]. The binary which has marks on each entry becomes the Nash equilibrium points. In this case,

$$SK_{start} = \begin{bmatrix} & Q & Q^{'} \\ Q & (0.9\bar{8}40, 0.9\tilde{8}40) & (0.9\bar{8}40, -0.0160) \\ Q^{'} & (-0.0160, 0.9\tilde{8}40) & (-0.0320, -0.0320) \end{bmatrix}$$

Finally, we see that there are two marks on the first binary of the bimatrix that is $(0.9840, 0.9840)$. In other words, the Nash equilibrium of this game is (Q,Q) strategy. It means that it is the best outcome for each player. Therefore, we conclude that staying in the quarantine during very beginning of the spread is the best option for each player. On the contrary, it is clear that the players have the worst outcome in the $(Q^{'},Q^{'})$ strategy as $(-0.0320, -0.0320)$. It is obvious that, $(Q^{'},Q^{'})$ strategy is definitely not worth to play.

As the second stage, we analyze the time period when the average of the test numbers and diagnoses reach the peak, that happens between the dates 26.02.2020 and 11.03.2020, that is referred as the spread. Similar to the previous case, we create the payoff bimatrix by follow up the payoff matrix creation steps for this stage as:

$$SK_{spread} = \begin{bmatrix} & Q & Q^{'} \\ Q & (0.9634, 0.9634) & (0.9634, -0.0366) \\ Q^{'} & (-0.0366, 0.9634) & (-0.0732, -0.0732) \end{bmatrix}$$

The Nash equilibrium point is $(0.9634, 0.9634)$, that is (Q, Q), for the spread stage. Even though the risk of the getting virus is increased, it is still best choice to stay inside. On the other hand, it is very important to notice that the loss of each player in (Q', Q') strategy is doubled comparing with the value in the start stage. Therefore, we may say that it becomes more risky to go out during the dates of peak.

As the last stage, we use the data from 12.02.2020 to 09.04.2020 the situation of the slowing down of the spread and we name this stage as end. In other words, we investigate the case during the government takes control over the spread. The payoff bimatrix is similarly generated as follows:

$$SK_{end} = \begin{bmatrix} & Q & Q' \\ Q & (0.9870, 0.9870) & (0.9870, -0.0130) \\ Q' & (-0.0130, 0.9870) & (-0.0260, -0.0260) \end{bmatrix}$$

We see that the Nash equilibrium point implies the same result that is staying inside with the payoffs $(0.9870, 0.9870)$. Moreover, we observe the loss of players in the breaking quarantine strategy (Q', Q') is decreased comparing to the spread stage. However, it is still clearly the worst option to be chosen by the players.

Furthermore, we examine the situations if the players have to make selection again, for instance, that is the players have to make a choice during the spread stage again. In other words, we consider the game as the repeated game in this situation. Here, we give the game tree of the spread stage as below, (see for the repeated game and the game tree in details [107]).

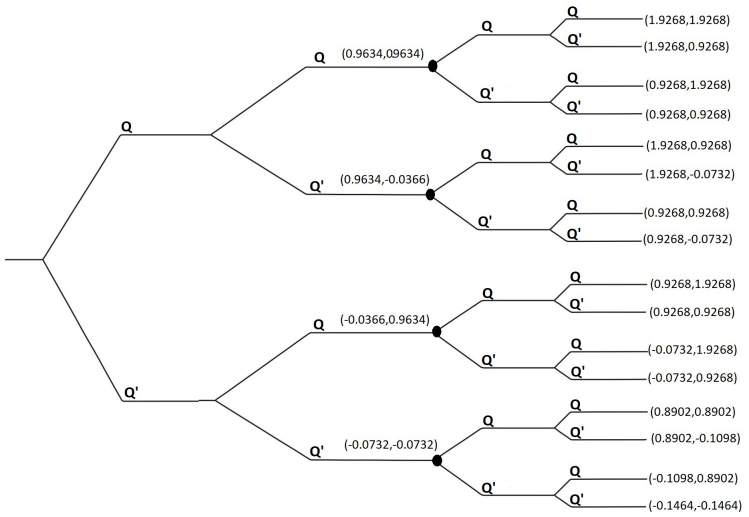


Figure 2.3 : Game tree for the second round for the spread

After that, we generate the payoff bimatrix for the second round of the game by using the values in the game tree as,

$$SK_2 = \begin{bmatrix} (1.9\bar{2}68, 1.9\tilde{2}68) & (1.9\bar{2}68, 0.9268) & (1.9\bar{2}68, 0.9268) & (1.9\bar{2}68, -0.0732) \\ (0.9268, 1.9\tilde{2}68) & (0.9268, 0.9268) & (0.9268, 0.9268) & (0.9268, -0.0732) \\ (0.9268, 1.9\tilde{2}68) & (0.9268, 0.9268) & (0.9202, 0.8902) & (0.8902, -0.1098) \\ (-0.0732, 1.9\tilde{2}68) & (-0.0732, 0.9268) & (-0.1098, 0.8902) & (-0.1464, -0.1464) \end{bmatrix}$$

The Nash equilibrium point is $(1.9268, 1.9268)$ which is the value of the strategy (Q, Q). Therefore, it can be seen that the player payoffs increased if they stay in the quarantine when the players have to make selection again. However, if the players go out repeatedly, their loss gets doubled, $(-0.1464, -0.1464)$ in this case. In other words, the probability of being infected doubled. Consequently, if the number of the play increases, the importance of the quarantine effects reflect more explicitly and also the loss and profit increase, relatively.

Case II: Italy

In this section, we investigate the effects of the quarantine during the first wave of Covid-19 pandemic in Italy as it is analyzed for South Korea. In order to do the investigation, we use the data from 21.02.2020 to 14.04.2020 [112]. The time period between 21.02.2020 and 06.03.2020 is named as the start. In addition to this, the spread got faster starting from 07.03.2020 until 14.04.2020 according to the data provided in [112] so that we refer this time period as the spread.

We use the same approach as it is done for the analysis of South Korea, and we begin the analysis with the first stage. We follow up the procedure given above in order to create the payoff matrix for the start case. The possibility of infection and not getting infected by the virus are evaluated as 0.1142 and 0.8858 by Equation 2.8, respectively. Then, we obtain the payoff bimatrix of the start case as:

$$It_{start} = \begin{bmatrix} & Q & Q' \\ Q & (0.8858, 0.8858) & (0.8858, -0.1142) \\ Q' & (-0.1142, 0.8858) & (-0.2284, -0.2284) \end{bmatrix}$$

We find the Nash equilibrium point as $(0.8858, 0.8858)$, that is, (Q, Q). We see that keeping the quarantine is the most advantageous choice that can be selected by the players for Italian people. However, we notice that Italy's payoff of the quarantine is worse than the payoff of South Korea, $(0.9840, 0.9840)$. Therefore, we can say that the

first stage in Italy is more critical than South Korea. Moreover, it can be seen that the loss for $(Q^{'},Q^{'})$ strategy of Italy is 7 times more than the loss of South Korea, which implies that the pandemic in Italy started worse than South Korea.

As the second stage, we analyze the spread case. At this stage, the possibility of getting the virus is 0.1849 and not getting the virus is 0.8151, which are evaluated by Equation 2.8. The payoff bimatrix is generated by the procedure as,

$$It_{spread} = \begin{bmatrix} & Q & Q^{'} \\ Q & (0.8151, 0.8151) & (0.8151, -0.1849) \\ Q^{'} & (-0.1849, 0.8151) & (-0.3698, -0.3698) \end{bmatrix}$$

The Nash equilibrium is also (Q,Q) strategy with the payoffs $(0.8151, 0.8151)$. It is clear that the payoffs in the second stage is decreased comparing to the first stage. The loss of $(Q^{'},Q^{'})$ strategy, i.e. breaking the quarantine, is increased more than 50%. Thus, we may say that breaking the quarantine when the spread rate is high becomes 50% more risky.

In addition to these analysis, we investigate the results if the players have to make selection for a second time during the spread stage. In other words, we convert the game to a repeated game same as it is done in the analysis of South Korea. We create the second round payoff bimatrix It_2 for Italy by using It_{spread} bimatrix as,

$$It_2 = \begin{bmatrix} (1.6\bar{3}02, 1.6\tilde{3}02) & (1.6\bar{3}02, 0.6302) & (1.6\bar{3}02, 0.6302) & (1.6\bar{3}02, -0.3698) \\ (0.6302, 1.6\tilde{3}02) & (0.6302, 0.6302) & (0.6302, 0.6302) & (0.6302, 0.3698) \\ (0.6302, 1.6\tilde{3}02) & (0.6302, 0.6302) & (0.4453, 0.4453) & (0.4453, -0.5547) \\ (-0.3698, 1.6\tilde{3}02) & (-0.3698, 0.6302) & (-0.5547, 0.6302) & (-0.7369, -0.7396) \end{bmatrix}$$

We obtain the Nash equilibrium point as $(1.6302, 1.6302)$, that is $(QQQQ)$ strategy, if we find the equilibrium points as we did for the previous illustrations. On the other hand, it is clearly seen that the worst strategy is $(-0.7369, -0.7396)$ which is $(Q^{'}Q^{'}Q^{'}Q^{'})$. If we compare the results with South Korea's results, we see that breaking the quarantine is 5 times more dangerous when the rate of the virus spread is high. Therefore, we may say that the Covid-19 crisis in Italy is worse than South Korea.

Case III: Türkiye

In this analysis, we consider Türkiye's data set starting from the date the very first

Covid-19 diagnosis, 11.03.2020, and the last data declared by the ministry of health in 03.05.2020 [113]. We follow up the same procedure to analyze the data as we did in the previous cases. We refer to the days between 11.03.2020 and 25.03.2020 as the start and the time period between 26.03.2020-03.05.2020 is named as the spread for Türkiye.

The payoff matrix generated by using the average of the tests/diagnoses calculated by Equation 2.8 for the first stage is below, and then we put a mark on the binaries to find Nash equilibrium point in the payoff matrix,

$$Tur_{start} = \begin{bmatrix} & Q & Q^{'} \\ Q & (0.9\bar{5}02, 0.9\tilde{5}02) & (0.9\bar{5}02, -0.0498) \\ Q^{'} & (-0.0498, 0.9\tilde{5}02) & (-0.0996, -0.0996) \end{bmatrix}$$

The Nash equilibrium is the same as the previous examples which is $(0.9502, 0.9502)$ or equivalently (Q, Q) strategy. It is clearly better than the same stage of Italy and slightly worse than South Korea. However, the comparison of the worst strategies, that is $(Q^{'}, Q^{'})$ strategy, shows that Türkiye is better than Italy and worse than South Korea.

As the second stage, we obtain the following payoff matrix for the spread stage,

$$Tur_{spread} = \begin{bmatrix} & Q & Q^{'} \\ Q & (0.8\bar{7}35, 0.8\tilde{7}35) & (0.8\bar{7}35, -0.1265) \\ Q^{'} & (-0.1265, 0.8\tilde{7}35) & (-0.2530, -0.2530) \end{bmatrix}$$

The equilibrium point is $(0.8735, 0.8735)$ which is the same strategy as in the previous cases. Even though it is worse than South Korea's payoff, it is better than Italy's payoff. Therefore, we may conclude that the pandemic in Türkiye expands slower than Italy and faster than South Korea.

As a final analysis, we generate a repeated game, as it is done in Case I and Case II above, for Türkiye by using Tur_{spread} data set. Then, we have the payoff matrix of the repeated game as below,

$$Tur_2 = \begin{bmatrix} (1.7\bar{4}70, 1.7\tilde{4}70) & (1.7\bar{4}70, 0.7470) & (1.7\bar{4}70, 0.7470) & (1.7\bar{4}70, -0.2530) \\ (0.7470, 1.7\tilde{4}70) & (0.7470, 0.7470) & (0.7470, 0.7470) & (0.6205, -0.3795) \\ (0.7470, 1.7\tilde{4}70) & (0.7470, 0.7470) & (0.6205, 0.6205) & (0.6205, -0.3795) \\ (-0.3795, 1.7\tilde{4}70) & (-0.3795, 0.6205) & (-0.3795, 0.6205) & (-0.5060, -0.5060) \end{bmatrix}$$

As in the start case, we find the pure Nash equilibrium point as $(1.7470, 1.7470)$. We can point out that the result is better than Italy and worse than South Korea.

In the start stage of the pandemic, South Korea achieved the highest payoff at 0.9840. Among Italy and Türkiye, the latter country recorded a higher payoff of 0.9205 compared to Italy's 0.8858, despite both countries having similar data sets. The (Q,Q) strategy, indicating continued quarantine, is clearly the optimal action for all countries during this stage. Notably, the (Q',Q') strategies reflect the pandemic's severity in each country. Italy's payoff of -0.2284 is the lowest, twice as detrimental as Türkiye's -0.0996 and over seven times worse than South Korea's -0.0320. This suggests that the pandemic escalated more rapidly in Italy than in the other two countries. During the second stage of the pandemic, characterized by increased case numbers, payoffs for all countries decline, indicating a heightened risk of infection. South Korea maintains the highest payoff at 0.9634, followed by Türkiye at 0.8735, while Italy continues to have the lowest payoff at 0.8151. The optimal strategy for all countries remains (Q,Q). In the context of the (Q',Q') strategy, Italy's payoff of -0.3698 is five times worse than South Korea's -0.0732 and 1.5 times worse than Türkiye's -0.2536. In the third stage, focusing solely on South Korea, the payoff for the (Q,Q) strategy has increased to 0.9870, surpassing the first stage payoff of 0.9840. This improvement likely reflects South Korea's effective control over the pandemic during this period.

In the second stage, when any player disregards precautions, the Nash equilibrium remains at (Q,Q,Q,Q), indicating that choosing to stay in quarantine is consistently the best option. The payoffs for the (Q,Q) strategy double to 1.9268 for South Korea, 1.6302 for Italy, and 1.7470 for Türkiye. However, the potential losses from breaking quarantine increase, heightening the risk of infection. Italy experiences the worst payoff at -0.7369, while South Korea achieves the best payoff at -0.1469, with Türkiye following at -0.5060.

To analyze the situation from a different angle, we calculate the death rates over diagnosed cases for each country using the formula $\lambda_{CountryName} = \frac{TotalNumberofDeaths}{TotaNumberofDiagnosedCase}$. The results are $\lambda_{SouthKorea} = 0.0196$, $\lambda_{Italy} = 0.1296$ and $\lambda_{Trkiye} = 0.0270$. These rates can be ranked as $\lambda_{SouthKorea} \leq \lambda_{Trkiye} \leq \lambda_{Italy}$. While death rates are influenced by various factors beyond quarantine measures-such as healthcare infrastructure-the ranking aligns with the quarantine requirements identified through our game-theoretical approach.

Figure 2.4 illustrates the cumulative deaths and diagnoses for each country. The death curve for Italy significantly diverges from those of South Korea and Türkiye. Notably, despite similar total diagnoses in Italy and Türkiye, Türkiye's death toll is considerably lower. Clearly, South Korea emerges as the most effective country in managing both deaths and cases, as indicated by the minimal numbers depicted in Figure 2.4.

We present the number of deaths and cases for South Korea, Italy, and Türkiye in Figure 2.4. We see that the number of cases in South Korea is extremely small compared to Italy and Türkiye. Additionally, we barely notice the death curve of South Korea since the number is very small. On the other hand, the comparison of the death

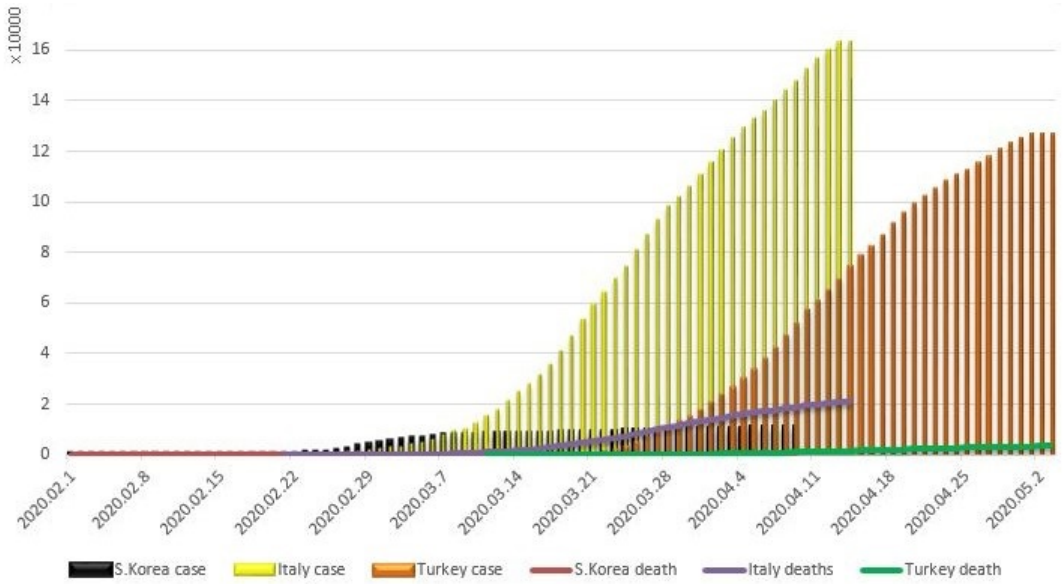


Figure 2.4 : The cumulative deaths and cases of South Korea, Italy and Türkiye.

curves of Italy and Turkey demonstrates that the control over the disease is better in Türkiye since its death curve is quite under the curve of Italy even though the numbers of the cases in both countries are similar.

The total number of diagnoses, deaths, and rate of deaths of some developed and developing countries are shown in Table 2.1. We can see that most of the developed countries are placed in the last lines of the table except South Korea. The first places belong to developing countries which is an unexpected situation.

In addition to these, we consider the infection rates before and after Covid-19 vaccination [114]. The results reveal that countries which began vaccination early and have a high number of vaccinated individuals should have a higher likelihood of not becoming infected. Although Türkiye started to vaccinate the people later than

Table 2.1 : The data for some developed vs developing countries on 03 May 2020

Countries	# of Total Diagnosis	# of Total Deaths	The rate of Deaths/Diagnosis
Russia	134687	1280	0.0095
Pakistan	20084	4576	0.0228
South Korea	10793	252	0.0233
Türkiye	126045	3397	0.0270
India	42505	1391	0.0327
Germany	167007	6866	0.0411
U.S.A.	1188112	68597	0.0577
Canada	59474	3682	0.0619
Iran	97424	6203	0.0638
Brazil	101147	7025	0.0695
Italy	210717	28884	0.1371
U.K.	186599	28446	0.1370

Israel, the U.S., the U. K. and Russia, Türkiye's performance outperformed the other countries due to the precautions taken earlier by the Turkish government [114].

3. MACHINE LEARNING APPLICATION FOR LARGE SCALE MATRIX GAMES

In this chapter, we introduce a novel machine learning-driven framework designed to solve large-scale zero-sum matrix games by leveraging insights gained from the offline extended matrix norm method [43] presented in Chapter 2 . While modern game-theoretic tools like the extended matrix norm method enable rapid game value estimation for small-scale zero-sum games through payoff matrix norms, accuracy diminishes as the number of strategies increases. To address this challenge, we propose a new neural network architecture specifically tailored for large-scale zero-sum matrix games. Firstly, we develop and train a neural network arhcitecture using data obtain by MN method for small scale-matrix games to see the efficiency of the neural network on the method [115]. Then, we improve and adapt this architecture takes the output estimations from the extended matrix norm method and the payoff matrix as inputs, providing quick estimations of the game value and proper strategy set as outputs. We train the model using a variety of randomly generated zero-sum games of different sizes. Results demonstrate that our framework achieves accurate value predictions with less than 10% absolute relative error for games featuring up to 50 strategies. Additionally, once trained, the network can deliver solution predictions in real-time, enhancing its applicability in real-world scenarios.

3.1 Machine Learning Supported the EMN Method for LS-ZS Matrix Games

Analytic solutions for large-scale zero-sum (LS-ZS) matrix games are generally more challenging to obtain than for small-scale zero-sum (SS-ZS) games due to the inherent complexity of their structures. Consequently, LS-ZS matrix games are often not preferred for modeling real-world problems, despite their potential for better representation compared to SS-ZS games. To address this complexity, researchers have begun exploring dimension reduction and other numerical techniques for LS-ZS matrix games. For example, Li et al. utilized dimension reduction to model a 16×12 matrix for air vehicle combat, comparing their results with various methodologies

[116]. While the extended matrix norm method is beneficial for both zero-sum and non-zero-sum matrix games [1, 42], it proves less practical for LS-ZS games due to the challenges in distributing elements of the mixed strategy set effectively. As payoff matrix size increases, the distribution of strategy elements becomes more complex, necessitating greater computational resources. To overcome these challenges, we propose integrating artificial intelligence with the EMN method, aiming to simplify the analysis of LS-ZS matrix games. This combination seeks to enhance computational efficiency and improve solution accuracy.

We first develop a neural network designed to predict the optimal distribution of elements within the mixed strategy set, as derived from the extended matrix norm method in Chapter 2. To facilitate these predictions, we input only the extrema of the strategy set into the network, allowing it to effectively allocate the elements within the set. Essentially, the neural network is employed to identify an appropriate strategy set based on the insights provided by the EMN method.

Here, our proposed architecture is displayed in Fig. 3.1 [43]. A forward pass through

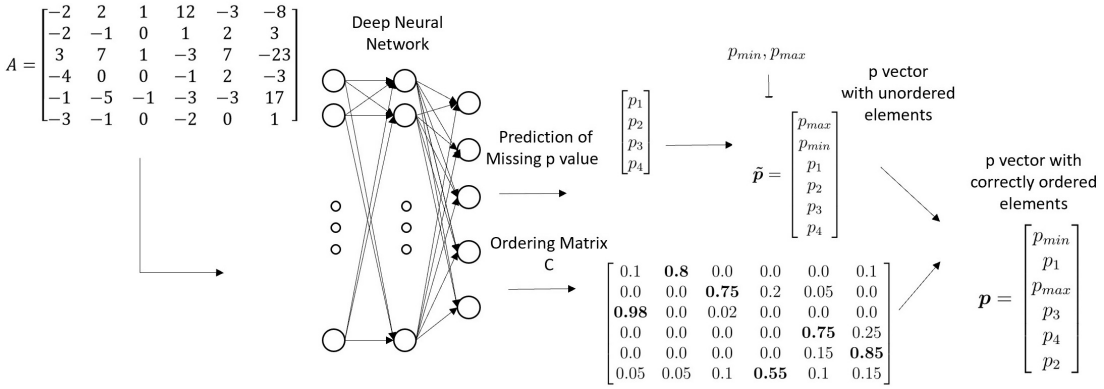


Figure 3.1 : Neural network architecture for prediction of missing p values.

the network work as follows; first the payoff matrix of the game is provided as an input. Network processes the payoff matrix using convolutions layers and projects it into a latent space vector. Next this latent vector is concatenated with the p_{min} and p_{max} values predicted with the EMN algorithm, and the network passes this information from several dense and convolutions layers to predict two outputs. First output is the missing p values, and the second outputs is a probability matrix that predicts the probability of location of each of the missing values. By taking the maximum column

element of this matrix, it is possible to reconstruct an approximate p vector, which can then be used to estimate an approximate value of the game.

3.1.1 The integration of machine learning to the EMN method

In this section, we outline the algorithm corresponding to our proposed method. We start by detailing the matrix game generation algorithm, followed by the algorithm for our neural network architecture. We also provide a brief overview of our database.

We present a comprehensive analysis of the solution performance across various matrix game dimensions and evaluates the neural network architecture's effectiveness. We begin with Algorithm 2, which is designed to generate random matrix games of different sizes and solve them. This algorithm also incorporates the EMN method to identify the extrema of the strategy sets obtained by Theorem 2.1.5 in Chapter 2 for the respective matrix games, thereby generating a dataset for training the neural network model [43].

Algorithm 2 Matrix Game Data Set Generator and conducting of the EMN method

for $i = 1, 2, \ldots, N$ **do**
    Generate $A = randi([x\ y], m, n)$
    Solve the matrix A $game\_solve(A)$ and store in an array
    Evaluate the related norms and store in an array
    while $tol > 0.01$
    Compute v_lower, v_upper, and store the v_dummy in the array $dummystore$
    Calculate p_min and p_max, and store in the array $strategy\ set$
    $tol = abs(dummystore(end) - dummystore(end-1))$
    end while
end for

where N represents the number of the game desired, x and y are positive integers with $x < y$, m and n denotes the size of the matrix.

By conducting Algorithm 2, we generate and consider three different sizes of matrix game, 5×5 (small scale), 10 (medium scale) and 50×50 (large scale). For each size, we generate up to 5×10^5 random payoff matrices. We reserve 80% of the sampled data to train the neural network model and use the 20% of data to check the model's performance on unseen data. Moreover, architectural details of the neural network portrayed in Fig. 3.1 is provided as follows [43]:

- The $N \times N$ payoff matrix is first normalized so that the training data set has zero mean and unit variance. Next, 3 stacked convolutional layers are applied on the

matrix data, where each layer has 3×3 filters and ReLu activation function given in Section 1.2.3. The number of filters in each layer are $64, 128, 256$ correspondingly. MaxPooling is applied in between layers 1 and 2. After layer 3, a flattening operation is applied to transform the output of the layer 3 into a 1D vector.

- The output of the convolutional layers is passed through 2 dense layers, with output sizes $64, 32$. These layers also have ReLU activation.

- Next, the output of dense layers is fed to two different heads, one for predicting the missing values of the p vector, and the other for estimating the relative order of the elements of the p vector. The first head is a dense layer with $N-2$ output size and linear activation. The second head is a dense layer with $N \times N$ output size, where each output column is passed through a softmax activation given in Chapter 1.2.3.

- In the final layer, predicted $N-2$ elements are fused with p_{min}, p_{max} from the data set (note that these values are written to the database by the EMN algorithm), and argmax layer is applied upon the predicted order matrix to re-order the elements of the fused p vector. The predicted p vector can be used for estimating the value of the game. Finally, loss per sample is computed by calculating the mean square error between the predicted game value and the ground truth game value.

For training the network, we used the Adam optimizer with an initial learning rate 10^{-3} and batch size of 32, and a validation split of 10%. Each training instance was trained for 1000 epochs, however, learning is terminated when the validation loss did not improve for 20 epochs.

3.1.2 Machine learning applications with the extended matrix norm method

In this subsection, we present the results obtained by the proposed neural network and investigates the performance of the neural network with respect to the mean square errors.

We present Table 3.1 demonstrating the game value prediction performance across different game and dataset sizes by the proposed neural network. It can be seen that neural network model's generalization performance improves greatly as the number of training instances increase. In particular, for small and medium instance, network

predicts the value of the game within $5\% - 6\%$ of the actual game value. For the LS-ZS matrix games, the average error is around 10%, which is still a remarkable performance given the size of these games [43].

Table 3.1 : Value prediction error of the developed models across different game sizes and number of training samples

Number of Games Used for Model Development	Value Prediction Error on Training Games (80% of all Games)			Value Prediction Error on Test Games (20% of all Games		
Size of Games	5×5	10×10	50×50	5×5	10×10	50×50
500	4.20%	4.90%	7.30%	33.20%	36.30%	40.30%
1000	4.70%	5.70%	6.20%	10.10%	12.10%	32.10%
2000	5.10%	6.10%	10.10%	6.20%	8.10%	28.10%
10000	5.20%	8.20%	11.90%	5.60%	9.10%	14.70%
20000	5.30%	9.30%	13.10%	5.50%	9.70%	14.20%
50000	5.30%	7.20%	10.30%	5.50%	8.40%	11.50%
100000	4.40%	6.40%	8.40%	5.20%	7.90%	10.20%
200000	4.60%	5.60%	8.20%	5.00%	6.10%	9.90%
500000	4.80%	5.80%	8.30%	4.90%	5.50%	9.90%

It can be observed that as the size of the games increases, average prediction error on both training and test sets get worse, since the model needs more samples to fit to the data on higher dimensional problems. On the other hand, for a fixed game size, increasing the number of samples improves the error rate on the test set significantly, whereas the error rate on the training set only increases slightly. These results reinforce that there is indeed a learnable pattern between the structure of the payoff matrix and the game value [43].

It can be seen that error on the test set can be significantly reduced by increasing the number of samples (i.e. number of randomly generated payoff matrices), and the error reduction rate depends strongly on the size of the game. Larger games require more samples to close the gap between training and test error. Graphs in Figure 3.2 proved an overview of how percentage error changes as a function of the number of training samples, for different game sizes. It can be observed from these figures that all error rates decay consistently as the number of samples is increased, which supports our hypothesis that a strong relationship exists between the actual game value, pay-off

matrix, and the p values computed by the EMN algorithm, which can be discovered by training machine learning algorithms from past solutions. Secondly, we can see that the decay rate of the error depends on the game size, as expected the bigger the size of the game, it takes more samples to bring down the error rate to the desired level [43].

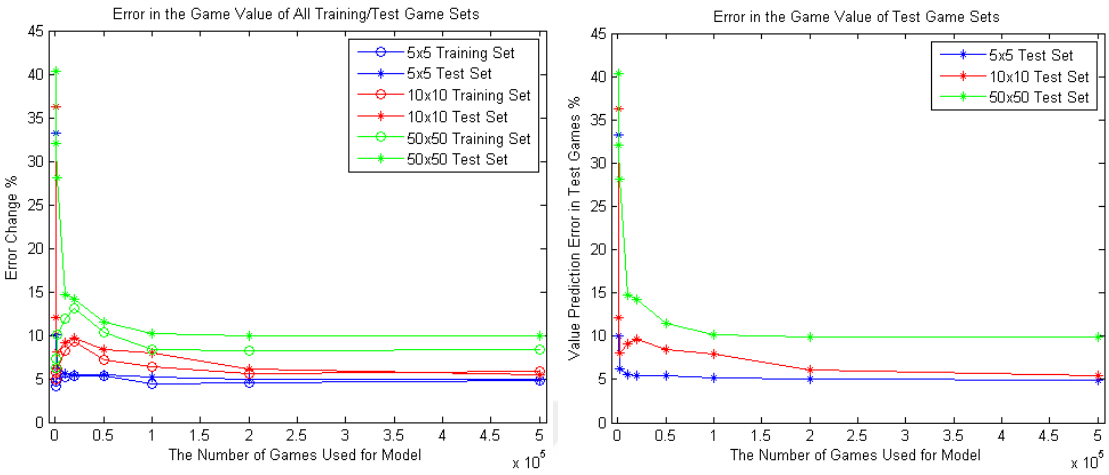


Figure 3.2 : Changes in the value prediction error vs. number of samples used in model development across training and test sets.

Note that one of the main contributions of our work is, once the neural network is trained for a specific problem size, the solution time becomes constant in the sense that neural network (coupled with EMN algorithm) will always predict the solution to the problem using the same amount of computation. This is in contrast with some of the existing works for solving large-scale games, such as [116], where the solution time can change significantly based on the structure of the problem.

4. HYBRID SHAPLEY AND ITERATIVE METHODS FOR STOCHASTIC GAMES

In this chapter, we introduce four alternative solution methods to Shapley iteration in Theorem 1.1.15 for solving stochastic matrix games presented in Definition 1.1.13 in Chapter 1. We begin by integrating the extended matrix norm method with Shapley iteration, subsequently formulating and proving both weak and strong hybrid versions of Shapley iterations. Next, we present the semi-extended matrix norm and iterative semi-extended matrix norm methods, which do not require analytic solutions and can find approximate solutions for stochastic matrix games without determining strategy sets. We provide comparisons among Shapley iteration, the hybrid Shapley iterations, and the semi-extended methods using several examples. The results demonstrate that both the weak and strong hybrid Shapley iterations enhance the original Shapley iteration, reducing the number of required iterations. Notably, the strong hybrid Shapley iteration outperforms all other proposed methods. Finally, we analyze the performance of these methods for large-scale stochastic matrix games, highlighting their effectiveness in practical applications.

4.1 Alternative Solution Methods for Stochastic Matrix Games

We propose four distinct alternative solution methods for stochastic matrix games. The traditional Shapley iteration method necessitates an analytical solution for each state, which can be difficult for large-scale games. To address this, we introduce weak and strong hybrid Shapley iterations aimed at reducing the number of iterations required in the solution process.

4.1.1 Hybrid Shapley iterations

In this subsection, we present hybrid iterative methods that consist of the EMN method given in Chapter 2 and Shapley iteration presented in Theorem 1.1.15. We begin with stating the weak hybrid Shapey iteration in the next theorem.

Theorem 4.1.1 ***(Weak Hybrid Shapley Iteration) [49]*** *For $k = 1,...,N$, each game G^k has a approximate game value $\bar{v}(k)$ obtained by v_{EMN}, and game value $v(k)$. These values are*

$$\begin{aligned} \bar{v}_n(k) &= v_{EMN}\left(a_{ij}^k + \sum_{l=1}^{N} P_{ij}^k(l) v_n(l)\right) \\ v_{n+1}(k) &= val\left(a_{ij}^k + \sum_{l=1}^{N} P_{ij}^k(l) \bar{v}_n(l)\right) \end{aligned} \tag{4.1}$$

the unique solution of the equations system, and the iterations for $n = 0,1,...$ are convergent for arbitrary initial values.

Proof. The proof is similar to the proof of Shapley iteration given in [53]. Let $T : \mathbb{R}^N \to \mathbb{R}^N$ and $Tx = y$ be a mapping for each $k = 1,...,N$, $y_k = \bar{v}_{EMN}(A^{(k)}(v))$ and $A^{(k)}(v) = a_{ij}^{(k)} + \sum_{l=1}^{N} P_{ij}^l v(l)$. First, we show that $1-s$, where $s = \min_{i,j,k} s_{i,j}^{(k)} > 0$, is contraction factor for the contraction map T by considering the lemma for two matrices of the same dimensions in [53] under the norm of $||x|| = \max_k\{|x_k| : k = 1,...,N\}$ for the EMN method:

$$\begin{aligned} ||Tx - Tz|| &= \max_k |v_{EMN}(A^{(k)}(x)) - v_{EMN}(A^{(k)}(z))| \quad (\text{for any } x,z \in \mathbb{R}^N) \\ &= \max_k \left| v_{EMN}\left(a_{ij}^{(k)} + \sum_{l=1}^{N} P_{ij}^{(k)}(l)x(l)\right) - v_{EMN}\left(a_{ij}^{(k)} + \sum_{l=1}^{N} P_{ij}^{(k)}(l)z(l)\right)\right| \\ &\leq \max_{k,i,j} \left| \sum_{l=1}^{N} P_{ij}^{(k)}(l)x(l) - \sum_{l=1}^{N} P_{ij}^{(k)}(l)z(l)\right| \\ &\leq \max_{k,i,j} \left(\sum_{l=1}^{N} P_{ij}^{(k)}|x(l) - z(l)|\right) \\ &\leq (\max_{k,i,j} \sum_{l=1}^{N} P_{ij}^{(k)}(l))||x - z|| = (1-s)||x-z|| \end{aligned}$$

since $1-s < 1$ according to the probability that the game stops, and v_{EMN} converges to the actual game value as stated in [42], there exists a unique vector $\bar{v}$ such that $T\bar{v} = \bar{v}$, this finishes the proof. This part of the proof suggests that the EMN method can be used as an iterative method for the stochastic matrix games, too.

Similarly, the rest of the proof which includes the Shapley iterations for the stochastic matrix game updated by $\bar{v}$ such that $y_k = val(A^{(k)}(\bar{v}))$ and $A^{(k)}(\bar{v}) = a_{ij}^{(k)} + \sum_{l=1}^{N} P_{ij}^l \bar{v}(l)$

can be completed as follows:

$$
\begin{aligned}
||T\bar{x} - T\bar{z}|| &= \max_k |val(A^{(k)}(\bar{x})) - val(A^{(k)}(\bar{z}))| \quad (\text{for any } \bar{x},\bar{z} \in \mathbb{R}^N) \\
&= \max_k \left| val\left(a_{ij}^{(k)} + \sum_{l=1}^{N} P_{ij}^{(k)}(l)\bar{x}(l)\right) - val\left(a_{ij}^{(k)} + \sum_{l=1}^{N} P_{ij}^{(k)}(l)\bar{z}(l)\right) \right| \\
&\leq \max_{k,i,j} \left| \sum_{l=1}^{N} P_{ij}^{(k)}(l)\bar{x}(l) - \sum_{l=1}^{N} P_{ij}^{(k)}(l)\bar{z}(l) \right| \\
&\leq \max_{k,i,j} \left(\sum_{l=1}^{N} P_{ij}^{(k)}(l)|\bar{x}(l) - \bar{z}(l)| \right) \\
&\leq (\max_{k,i,j} \sum_{l=1}^{N} P_{ij}^{(k)}(l))||\bar{x} - \bar{z}|| = (1-s)||\bar{x} - \bar{z}||
\end{aligned}
$$

since $1 - s < 1$, and the analytic solution of each updated stochastic matrix game converges to the actual game value, there exists a unique vector v such that $Tv = v$, the proof is complete ■.

Theorem 4.1.2 ***(Strong Hybrid Shapley Iteration) [49]*** *For $k = 1,...,N$, each game G^k has a approximate game value $\bar{v}(k)$ obtained by v_{IEMN} iteratively, and game value $v(k)$. These values are*

$$
\begin{aligned}
\bar{v}_n(k) &= v_{IEMN}\left(a_{ij}^k + \sum_{l=1}^{N} P_{ij}^k(l)v_n(l)\right) \\
v_{n+1}(k) &= val\left(a_{ij}^k + \sum_{l=1}^{N} P_{ij}^k(l)\bar{v}_n(l)\right)
\end{aligned} \tag{4.2}
$$

the unique solution of the equations system, and the iterations for $n = 0,1,...$ are convergent for arbitrary initial values.

Proof. Here, the iterative extended matrix norm (IEMN) method represents the application of Theorem 2.1.5 - 2.2.2 recursively for a certain tolerance after applying the EMN method. Thus, the rest of the proof is similar to the proof of Theorem 4.1.1 which is based on the matrix norm method's convergence that is discussed in [42], and the convergence of the Shapley iteration. Therefore, it is omitted ■.

As a summary of the application process of the hybrid Shapley methods is presented as [49]:

Algorithm 3 Hybrid Shapley Iteration Methods

1: Select $v(0) = (0,0,...,0)$ (or an arbitrary vector) as an initial game value.
2: Find approximate game value $\bar{v}$ by Equation 4.1.1 for the weak hybrid Shapley iteration or, by Equation 4.1.2 for the strong hybrid Shapley iteration.
3: Update the stochastic matrix game with the game value obtained in Step 2.
4: Solve the updated matrix game analytically and plug the obtained game value v into the stochastic matrix game.
5: Repeat the process from Step 2 until a certain tolerance is reached for the game value.

4.1.2 Novel iterative methods

While the weak hybrid Shapley iteration (WHSI) and strong hybrid Shapley iteration (SHSI) reduce iteration counts, they still rely on analytical solutions. To eliminate the need for analytical solutions in the iterative process, we propose new iterations that utilize only the EMN or IEMN methods.

Theorem 4.1.3 *(**Semi-EMN Method for Stochastic Matrix Games** [49]) For $k = 1,...,N$, each G^k game has an approximate game value $\bar{v}(k)$ obtained by v_{SEMN} which is the mid-point of the boundaries obtained by Theorem 2.2.2. These values are*

$$\bar{v}_{n+1}(k) = v_{SEMN}\left(a_{ij}^{(k)} + \sum_{l=1}^{N} P_{ij}^{(k)}(l)\bar{v}_n(l)\right)$$

the unique solution of the equations system, and the iteration is convergent for arbitrary initial values.

Proof. The proof of this theorem parallels the methodology employed in the proof of Theorem 4.1.1. Thus, the details are left to the reader ■.

Theorem 4.1.4 *(**Iterative Semi-EMN Method for Stochastic Matrix Games** [49]) For $k = 1,...,N$, each G^k game has an approximate game value $\bar{v}(k)$ obtained by v_{ISEMN} which is the mid-point of the boundaries obtained by the EMN method iteratively. These values are*

$$\bar{v}_{n+1}(k) = v_{ISEMN}\left(a_{ij}^{(k)} + \sum_{l=1}^{N} P_{ij}^{(k)}(l)\bar{v}_n(l)\right)$$

the unique solution of the equations system, and the iteration is convergent for arbitrary initial values.

Proof. The proof of the theorem resembles the proof of Theorem 4.1.2. Thus, it is skipped ■.

Remark 4.1.5 *[49] Considering the fact that the Shapley iteration, WHSI, and SHSI need analytic solutions for the stochastic matrix games in the corresponding step of iterations, it is important to notice the semi-extended matrix norm (SEMN) method and iterative semi-extended matrix norm (ISEMN) method presented in Theorem 4.1.3 and Theorem 4.1.4 focus on only the approximate solution of the game and require no analytic solution in any step of the solution process. Therefore, in this perspective, this makes it easier to apply the methods compared to other methods. On the other hand, it is important to note that the EMN and IEMN methods are not used in the complete form, for simplicity, in the illustrations of the proposed methods. In other words, the game value chosen in the application of both the EMN and IEMN methods is the mid-point of the interval obtained in Theorem 2.2.2, i.e. the dummy or approximate game value, and no proper strategy set is created and corresponding approximate game value is calculated contrary to the use in [42, 43]. One can certainly obtain better results using the matrix norm-based methods in their complete form.*

4.2 Applications and Comparisons of the Alternative Methods

This section presents illustrative examples to demonstrate the applications and comparisons of the proposed methods for small-scale stochastic matrix games. Additionally, we provide simulation-based performance evaluations of these methods for larger-scale stochastic matrix games.

Example 1. [49,53] Let us consider the single-state stochastic matrix game G given as

$$G = \begin{bmatrix} 1+0.6G & 3+0.2G \\ 1+0.8G & 2+0.4G \end{bmatrix}$$

We solve this game by Shapley iteration, WHSI, and SHSI for 10^{-3} tolerance. We present the game values in each iteration in Table 4.1.

As shown in Table 4.1, the Shapley iteration converges at the 13th step, while both the weak hybrid Shapley iteration (WHSI) and strong hybrid Shapley iteration (SHSI) converge at the 8th step for the same tolerance level. Furthermore, we

Table 4.1 : The comparison of the results obtained at each step for 10^{-3} tolerance by the Shapley iteration, WHSI and SHSI.

#	**Shapley**	**WHSI**	**SHSI**
1	0	0	0
2	1	2.13000	2.14275
3	1.8	3.14592	3.16435
4	2.44	3.43107	3.44197
5	2.952	3.50095	3.50668
6	3.25486	3.51744	3.52117
7	3.40733	3.52128	3.52438
8	3.47854	3.52217	3.52510
9	3.51052		
10	3.52462		
11	3.53078		
12	3.53464		
13	3.53515		

consider the results from the Shapley iteration as reference values, given that it is a widely recognized and commonly used method for solving stochastic matrix games. Therefore, the relative errors are obtained as

$$error_{WHSI} = \frac{|v_{Shapley} - v_{WHSI}|}{v_{Shapley}} = 0.0036\ (\sim \%0.4),$$

$$error_{SHSI} = \frac{|v_{Shapley} - v_{SHSI}|}{v_{Shapley}} = 0.0028\ (\sim \%0.3).$$

These errors suggest that both hybrid iteration methods converge to game value and have good performance versus the Shapley method. Finally, the error between the WHSI and SHSI methods is revealed as

$$error = |v_{WHSI} - v_{SHSI}| = |3.52217 - 3.52510| = 0.00283.$$

The comparison between WHSI and SHSI reveals an absolute error of approximately 0.3%. Since WHSI employs the EMN method directly without iteration, this level of error is deemed acceptable. Although WHSI offers a slight reduction in computational cost, it incurs a marginally higher error compared to SHSI. In contrast, the iterative use of the EMN method in SHSI method leads to increased computational costs, but it provides results that converge more closely to the actual game value in fewer iterations. Thus, while a higher calculation cost enhances result certainty, both methods converge at the same step, allowing them to reach the game value with relative error much more quickly than the Shapley iteration.

Additionally, we also solve the illustrative example by using the SEMN and ISEMN methods in Theorem 4.1.3 and 4.1.4 for 10^{-3} tolerance, and summarize the results with the Shapley iteration in Table 4.2.

Table 4.2 : The comparison of the results obtained at each step for 10^{-3} tolerance by the Shapley iteration, SEMN and ISEMN.

#	**Shapley**	**SEMN**	**ISEMN**
1	0	0	0
2	1	1.4125	1.42844
3	1.8	2.34868	2.37665
4	2.44	2.87189	2.91263
5	2.952	3.15745	3.19752
6	3.25486	3.31098	3.34510
7	3.40733	3.39287	3.42063
8	3.47854	3.43666	3.45905
9	3.51052	3.46003	3.47855
10	3.52462	3.47246	3.48843
11	3.53078	3.47907	3.49343
12	3.53464	3.48258	3.49596
13	3.53515	3.48444	3.49724
14		3.48543	3.49789

It is seen that the results obtained by the SEMN and ISEMN methods are as good as the result of the Shapley iteration even though the SEMN and ISEMN converge at 14^{th} step. The relative errors of the methods are

$$error_{SEMN} = \frac{|v_{Shapley} - v_{SEMN}|}{v_{Shapley}} = 0.014\ (\%1.4),$$

$$error_{ISEMN} = \frac{|v_{Shapley} - v_{ISEMN}|}{v_{Shapley}} = 0.0105\ (\sim \%1).$$

While the SEMN and ISEMN methods converge at 14^{th} step, the ISEMN method provides a better approximate result. Moreover, the comparison of the SEMN and ISEMN methods is $|v_{SEMN} - v_{ISEMN}| = 0.01246$. Consequently, we compare the performances of the SHSI, WHSI, SEMN, and ISEMN methods on a 2×2 single-state stochastic matrix game concerning the relative error with the result of the Shapley iteration. SHSI and WHSI appear to perform better than all the proposed methods with %0.3 and %0.4 relative errors, respectively. However, the performance of SEMN and ISEMN methods can be acceptable since these are analytic-solution-free methods with less than %1.5 relative error. It is seen that the ISEMN outperforms the SEMN

method since it converges to the game value better than the SEMN method. When the performance order of the proposed methods is concerned as in Figure 4.1, it is in order of the SHSI, WHSI, ISEMN, and SEMN methods.

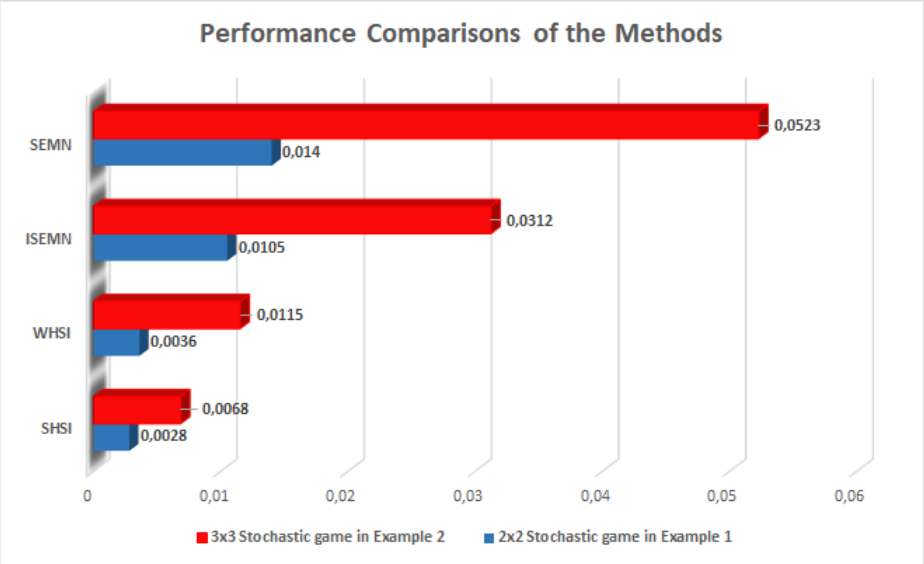


Figure 4.1 : Relative error comparisons of the proposed methods compared to the Shapley iteration method.

Example 2. [49] Consider the 3×3 single-state stochastic matrix game G given as

$$G = \begin{bmatrix} 6+0.0176G & 6+0.1409G & 2+0.1392G \\ 5+0.2460G & 8+0.1826G & 7+0.3277G \\ 5+0.0897G & 3+0.3142G & 9+0.1005G \end{bmatrix}$$

We solve the 3×3 single-state stochastic matrix games by using the Shapley iteration, SEMN, ISEMN, WHSI, and SHSI methods and present the obtained results in Table 4.3:

Table 4.3 : The comparison of the results obtained for 3×3 single-state stochastic matrix game at each step for 10^{-3} tolerance by the Shapley iteration, SEMN, WHSI, ISEMN, and SHSI

#	**Shapley**	**SEMN**	**WHSI**	**ISEMN**	**SHSI**
1	0	0	0	0	0
2	5.44444	6.03839	6.48544	5.53880	6.36254
3	6.33933	6.84862	6.69985	6.30243	6.57824
4	6.55947	6.95992	6.70710	6.40736	6.58552
5	6.61363	6.97524	6.70734	6.42177	6.58577
6	6.62695	6.97735		6.42375	
7	6.63023	6.97764		6.42402	
8	6.63104				

Before detailed comparison analyses, we calculate the relative errors of each proposed method concerning the Shapley iteration result as follows:

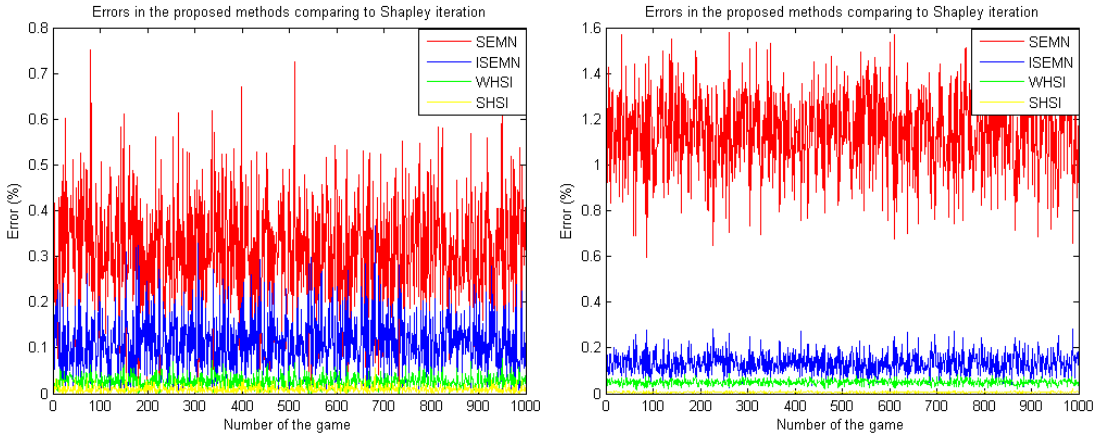
$$\begin{aligned} error_{WHSI} &\approx \%1, \quad error_{SHSI} \approx \%0.7 \\ error_{SEMN} &\approx \%5, \quad error_{ISEMN} \approx \%3. \end{aligned}$$

First of all, it is clear that the WHSI and SHSI methods approximate the result faster than the Shapley iteration by Table 4.3. However, the WHSI generates more errors than the SHSI since it uses the EMN method directly. The SHSI method exhibits the best performance among the proposed methods with a 0.0068 relative error. In addition to these, the SHSI converges faster than the Shapley iteration. In other word, the SHSI converges at 5^{th} iteration while the Shapley iteration converges at 8^{th} iteration.
If we compare the methods with no analytic solution, which are the SEMN and ISEMN methods, the ISEMN method approximates much better than the SEMN method with 0.03 relative error. Although these methods converge slower than the others, the solution process does not require the analytic solution, which is the most advantageous part of these methods.

If we examine how well the SHSI, WHSI, SEMN, and ISEMN methods perform concerning the relative error with the Shapley iteration result on 3×3 single-state stochastic matrix games, we notice that the SHSI and WHSI appear to outperform every suggested approach, with relative errors of $\sim$ %0.7 and %1, respectively. However, the ISEMN and SEMN methods have %3 and %5 relative errors, whose performances may be considered satisfactory due to no requirements of the analytic solution. It is important to notice that the ISEMN method converges to the game value more than the SEMN method, it is evident that it performs better than the latter in this example, too. The suggested methods' performance order is SHSI, WHSI, ISEMN, and SEMN methods as seen in Figure 4.1.

Figures 4.2a and 4.2b present the error analyses results for 5×5 and 10×10 games, respectively. To evaluate the performance of the proposed methods for single-state large-scale stochastic matrix games, we randomly generated and solved 1000 stochastic matrix games of sizes 5×5 and 10×10. Figure 4.2 illustrates the performance ranking of the methods. In Figure 4.2a, the proposed methods maintain the same order of performance observed in the previous example for 5×5 games.

Notably, the SEMN and ISEMN methods are distinctly separated from the SHSI and



(a) (b)

Figure 4.2 : Relative error comparisons of the proposed methods compared to the Shapley iteration method. (a) Error analyses results for 5×5 games (b) Error analyses results for 10×10 games

WHSI methods. The SHSI method demonstrates a significant advantage over the WHSI method. For 10×10 stochastic matrix games, the SHSI method continues to outperform WHSI, ISEMN, and SEMN methods, as depicted in Figure 4.2b. However, the SEMN method stands apart from the other proposed methods for this larger game size. The differences in performance among the proposed methods become more pronounced in larger-scale stochastic matrix games. Importantly, the ranking order of the methods remains consistent, following the sequence of SHSI, WHSI, ISEMN, and SEMN, similar to that observed in small-scale games.

5. A HOLISTIC ALTERNATIVE METHOD FOR STOCHASTIC GAMES: MARKOV REWARD GAMES

In this chapter, we explore single-agent stochastic games, particularly Markov reward games, given in Definition 1.1.16 of Chapter 1, represented as decision trees. We propose an alternative solution method utilizing matrix norms, distinguishing it from traditional state-and-action-based approaches like value iteration, policy iteration, and dynamic programming prensented in Section 1.1.4 of Chapter 1. Our matrix norm-based method addresses relevant stages and actions collectively, solving each stage holistically without the need to compute individual action effects on state rewards. This approach involves transforming the decision tree into a payoff matrix for each stage and applying matrix norms to the resulting matrices. We also integrate the concept of a moving matrix, given by Definition 1.1.17, to account for the simultaneous impacts of all actions at each stage, enhancing the method's holistic nature. An explanatory algorithm for implementation and a comprehensive solution diagram illustrate the method. By leveraging the simplicity of matrix norms, we present a novel perspective for solving these games. To clarify the matrix norm-based method, we provide a figurative application on a benchmark 2-stages, 2-actions Markov reward game, alongside a detailed implementation for a game with 3-stages and 3-actions.

5.1 Markov Reward Games and Matrix Norm-Based Method

The solutions to single-agent stochastic games, particularly Markov reward games, typically employ classical methods such as value iteration, policy iteration, and dynamic programming [62, 117, 118] preesented in Section 1.1.4 of Chapter 1 . In value iteration, the value function, which reflects the cumulative expected value from each state, is updated iteratively until convergence is achieved, after which the optimal policy is derived. Policy iteration begins with an initial policy and refines it through performance evaluations until an optimal policy is determined. Dynamic programming requires a tailored system for the problem, which can complicate its formulation

due to the absence of a universal mathematical structure; specific equations must be crafted for each unique situation [65]. All these approaches address the game on a state-by-state basis, evaluating the effects of each action separately. This method incurs additional computational overhead, as each requires an update process between states and a distinct evaluation of action effects, thereby extending the solution time. Notably, the lack of a compact, holistic approach for addressing single-agent stochastic games in tree form, which considers the collective impact of all actions at each stage, presents a significant gap in the literature [52].

To fill this gap, we propose a novel method that comprehensively tackles Markov reward games by treating all actions and states of a stage as a whole while incorporating matrix norms into the solution process. Our method begins by transforming the decision tree into a payoff matrix. Subsequently, we construct a moving matrix using both the payoff matrix and the transition matrix to evaluate the simultaneous effects of all actions at each stage. This moving matrix enables a collective assessment of the impact of all states on the rewards. We then utilize the ∞-norm of the moving matrix along with algebraic operations to generate a new matrix that represents optimal strategies, indicated by zeros in each column for the relevant stage. Following this, an update process leverages the 1-norm of the associated moving matrix and further operations to refine strategies for the remaining stages. This iterative process continues until an optimal strategy set is identified for all stages [52].

In the context of this thesis, we consider Markov reward games in the form of the decision tree which is generally depicted in Figure 5.1. The decision tree illustrated

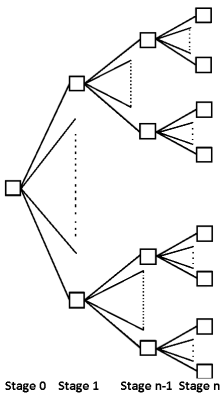


Figure 5.1 : A Markov reward game in decision tree form.

in Figure 5.1 consists of n stages, with each stage featuring n actions. The primary

objective of all solution methods is to identify an optimal strategy for each stage, or equivalently, an appropriate strategy set for each node in the diagram.

Typically, the resolution of a Markov reward game is accomplished through value iteration, policy iteration, and dynamic programming, which are briefly outlined in the following subsection. However, we proceed to introduce the concept of the moving matrix, along with the 1-norm and ∞-norm of a matrix defined in Section 1.1.5 of Chapter 1. These will serve as essential and differentiating tools in the proposed matrix norm-based solution method. In this approach, we incorporate the effects of all actions within a stage simultaneously by employing the moving matrix concept defined in Definition 1.1.17 given in Section 1.1.5.

5.1.1 An holistic alternative solution approach for single-agent stochastic games

In this section, we present the matrix norm-based method in addition to the transformation of the decision tree into payoff matrix form, which is the main step for the implementation of the solution method [52].

At the beginning of the matrix norm-based method, the decision tree of a Markov reward game is transformed into a payoff matrix for each stage, separately. For instance, the transformation for the n^{th} stage starts from its top reward until the last reward for each distinct node in the $(n-1)^{th}$ stage. These values are written as a column of the payoff matrix for each node of the $(n-1)^{th}$ stage. Then, the obtained payoff matrix for the corresponding stage is denoted as A_s for $s = 0, 1, ..., n$ which is depicted in Figure 5.2.

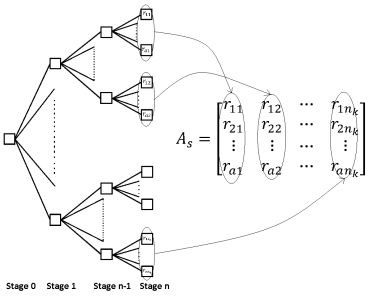


Figure 5.2 : The decision tree transformation to payoff matrix form.

Then, the column-wise moving matrix M_s^c is involved in the process to evaluate the impacts of all the actions on the corresponding rewards of the state at once contrary to the value iteration and policy iteration. In these methods, a similar evaluation is done

by using Equation 1.3 (value iteration) and Equation 1.4 (policy iteration), given in parts (a) and (b) of Section 1.1.4 for each state individually. For example, the moving matrix concept is equivalent to the part $\gamma P(s,a,s^{'})V(s^{'})$ given in Equation 1.3 , that is the value iteration. In order to obtain the moving matrix M_s^c, we multiply the payoff matrix γA_s at the corresponding stage by the transition matrix $P \in R^{a \times a}$ of a game with a actions, shortly $M_s^c = P(\gamma A_s)$, where P is defined as

$$P = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1a} \\ p_{21} & p_{22} & \cdots & p_{2a} \\ \vdots & \vdots & \cdots & \vdots \\ p_{a1} & p_{a2} & \cdots & p_{aa} \end{bmatrix}$$

where p_{ij} stands for the probability of transition from State i to State j and $i, j = 1, 2, ..., a$ [52]. Then, each window of the moving matrix M_s^c provides the rewards for each state considering all the actions. Next, a matrix S_s is created using the ∞-norm of each column of the moving matrix M_s^c to obtain the maximum reward for each state as in the *max* operator in Equation 1.3. Following that, a matrix St_s is generated computing the difference of the matrices M_s^c and S_s for simply signifying the optimal strategies for the corresponding stage with the places of zeros in each column, which completes the policy evaluation process for a stage. To move the previous stage since the method works backwardly, an update operation includes the solution process with an updated matrix A_{s-1}^u consists of 1-norm of the moving matrix M_s^c to add the total rewards in the Stage s to Stage $s-1$, as it is done by the addition operation for each state while using Equation 1.3 in Section 1.1.4 as $R(a,s) + \gamma \sum_{s'} P(s,a,s^{'})V(s^{'})$ for $\gamma = 1$ and similarly in Equation 1.3 , that is the policy iteration, since the columns of the moving matrix is established using the corresponding rewards in the stage s. The updating process is done as for each $j = 1, 2, \ldots, n_{k-1}$, where the reward values in matrix $A_{s-1}(i, j)$ are defined with $t = 0$, they are sequentially defined as $t = t + 1$ for all corresponding $i = 1, 2, ...a$ values. After updating as $A_{s-1}^u(i, j) = ||M_s^c(:, t)||_1 + A_{s-1}(i, j)$, the matrix is redefined as $A_{s-1}(i, j) = A_{s-1}^U(i, j)$. Finally, the Stage $s-1$ is updated and the process begins again and continues until the first stage. It can be observed that the proposed matrix norm-based method aligns with the general formulation of both value iteration and policy iteration. Consequently, these methods can be considered as dual to each other. The diagram in Figure 5.3 summarizes the solution process figuratively [52]. Additionaly, the algorithmic form of the matrix

norm-based method for the Markov reward game is given by Algorithm 4: Let a be the total number of actions, k be the index of the current node, n_k be the total number of payoff at $(k-1)^{th}$ stage and s be the index of the stage, $A_s \in \mathbb{R}^{a\times n_k}$ is the payoff matrix of the corresponding stage, $P \in \mathbb{R}^{a\times a}$ is the transition probability matrix. To find the strategy for the corresponding stage [52]:

Algorithm 4 Algorithm for the Matrix Norm-Based Method

1: Establish the matrix $A_s \in \mathbb{R}^{a\times n_k}$ as depicted in Figure 5.2.
2: Define the moving matrix $M_s^c = P(\gamma A_k) \in \mathbb{R}^{a\times n_k}$ for $\gamma \in [0,1]$.
3: Obtain the matrix $S_s \in \mathbb{R}^{a\times n_k}$ as $S_s(:,j) = ||M_s^c(:,j)||_\infty$ for $j = 1,2,...,n_k$.
4: Evaluate the strategy representative matrix $St_s \in \mathbb{R}^{a\times n_k}$ as $St_s = M_s^c - S_s$.
5: Update the rewards as $A_{s-1}^U(i,j) = ||M_s^c(:,t)||_1 + A_{s-1}(i,j)$.
6: Redefine $A_{s-1}(i,j)$ as $A_{s-1}^U(i,j)$ and repeat Step 1-6 for all stages.
7: Finally, the zeros in the matrix St_k signifies the optimal strategy for the game.

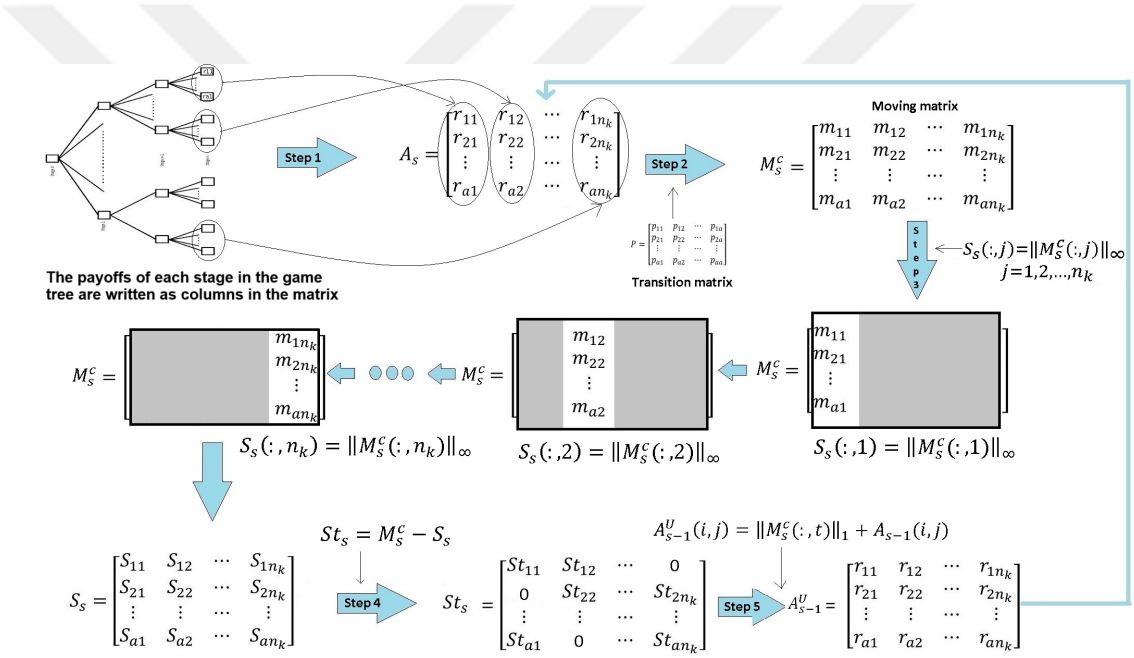


Figure 5.3 : The solution diagram for the matrix norm-based method.

The next remarks express some conditions on the structure of the decision tree for the smooth utilization of the matrix norm-based method.

Remark 5.1.1 *[52] The above algorithm should be applied for the shifted dual game that will be created by shifting all the entries in the tree by the amount of the absolute value of the smallest entry in the rewards if there are rewards with negative entries in the game tree. This operation is based on the fact that the game strategy is invariant under equal shifts of all rewards in the tree (equally shifted dual game).*

Remark 5.1.2 *[52] In case of multiple connections to the same node/point according to the tree structure of the game, attention should be paid to these repeating points in the matrix form after the reward value of the game is updated at each point where the reward values need to be updated backward.*

Remark 5.1.3 *[52] In case of perturbation of the rewards in the decision tree due to the reason in Remark 1, in other words, if there is a need for the generation of an equally shifted dual game, the actual game value (or reward) of each state can be evaluated by shifting back as follows: Let $r(S_k)$ represents the actual reward of the k^{th} state, $r(\bar{S}_k)$ denotes the perturbed reward of the k^{th} state, β_k refers the total number of distinct connection made by State k with forward states in all next stages (see Remark 5.1.1) and x denotes the amount of perturbation. Then, the actual reward of the k^{th} state is evaluated as $r(S_k) = r(\bar{S}_k) - (\beta_k + 1)x$.*

5.2 Illustrative Applications of Holistic Matrix Norm-Based Method

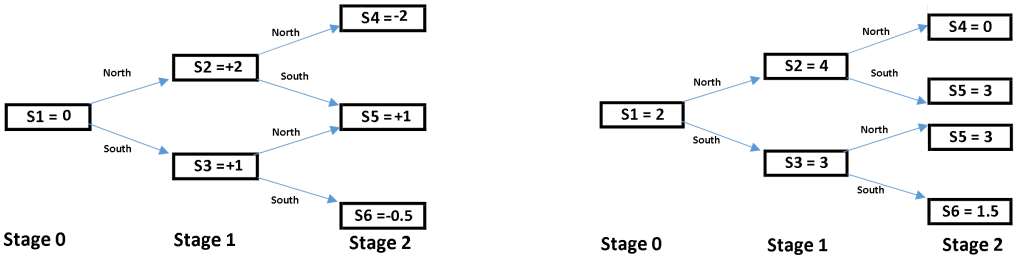
In this section, we provide two distinct examples to illustrate the application of the matrix norm-based method as an alternative to existing approaches in the literature. The first example will be solved figuratively using the flowchart presented in Figure 5.3, while the second will involve a direct implementation of the algorithm to showcase the proposed method from two different perspectives. For the sake of simplicity, we set $\gamma = 1$.

Example 1. [52,119] When the captain of the ship with a broken compass wants to go north, he has a 0.8 probability to go north and a 0.2 probability to go south. Similarly, when he wants to go south, he goes south with a probability of 0.8 and goes north with a probability of 0.2. The game tree of the modal game is presented in Figure 5.4a.

In the literature, these games are typically solved through dynamic programming, policy iteration, or value iteration, which have some challenges in the solution process mentioned in the introduction. In [119], the optimal strategy set is given $\{S,S,N\}$ for the problem.

The solution of the game will be obtained directly by using the solution diagram depicted in Figure 5.3 of the matrix norm-based method developed for the Markov reward games. To solve the game using the matrix norm-based method, it is necessary

to shift the game by +2 and the updated game tree is given in Figure 5.4b according to Remark 5.1.1.



(a) (b)

Figure 5.4 : The perturbed tree (b) is obtained by shifting all the rewards in the tree by the amount of the absolute value of the smallest entry in the original game tree (a).

The step-by-step application of the method is figuratively presented in Figures 5.5 and 5.6 as below. It starts with the transformation of the decision tree in Figure 5.4b to the form of a payoff matrix as depicted in Figure 5.2.

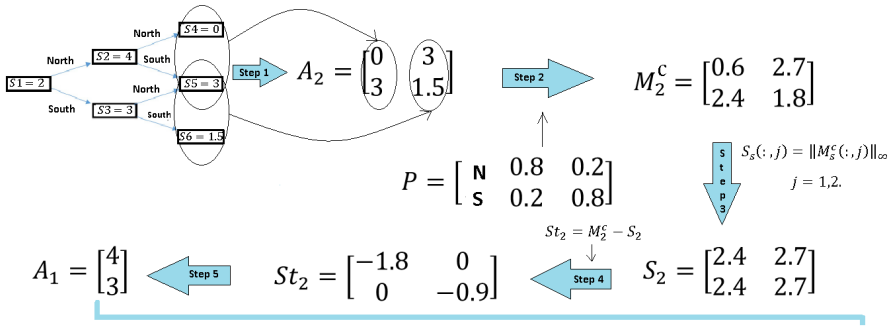


Figure 5.5 : Step-by-step illustration of the solution for stage 2.

In Figure 5.5, at Step 4, the optimal strategy for the 2nd stage is identified as {South, North}, as explained in the corresponding step of the solution diagram. Similarly, we update the matrix A_1 and proceed with the process as outlined below.

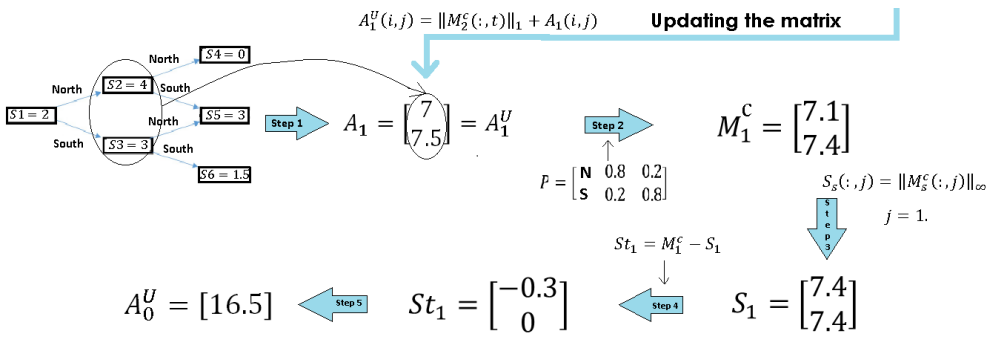


Figure 5.6 : Step-by-step illustration of the solution for stage 1.

In Figure 5.6, at Step 4, we observe that the zero in the matrix St_1 signifies that the optimal strategy is {South}. Consequently, the optimal strategy for the game is {South, South, North} which is the same as the result provided in [119]. We can also evaluate the reward of actual game for S_1 using Remark 5.1.3 as $r(S_1) = r(\bar{S_1}) - (\beta_1 + 1)2 = 16.5 - (6+1)2 = 2.5$, which is consistent with the result in [119]. Similarly, we can evaluate the actual reward of other states. For example, consider state S_2 and state S_4, $r(S_2) = r(\bar{S_2}) - (\beta_2 + 1)2 = 7 - (2+1)2 = 1$ and $r(S_4) = r(\bar{S_4}) - (\beta_4 + 1)2 = 0 - (0+1)2 = -2$, which are compatible with the results in [119]. It is important to notice that, the solution process utilizing the value iteration includes constant repetition of the process and advances state by state. It converges to the result in the third iteration as shown in Pirate Game in [119]. However, the matrix norm-based method does not require repeating the computation process to obtain the optimal policy and advance based on states but stages. It completes the solution process by considering each stage holistically, which is the most advantageous part of the method.

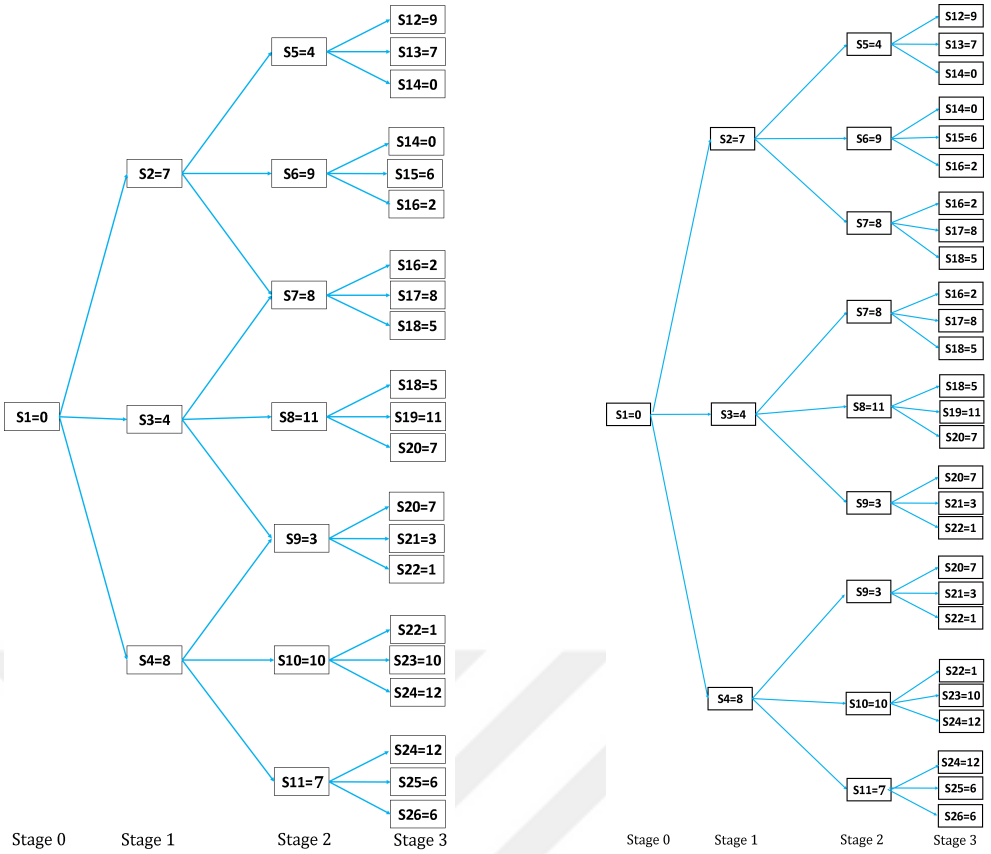
Example 2. [52] To illustrate the application of the matrix norm-based method using Algorithm 4 for Markov reward games with a relatively larger decision tree, we present the game with 3-stages and 3-actions (Up, Straight, Down) represented by the tree structure in Figure 5.7a. In order to apply the matrix norm-based method explicitly, we firstly obtain the equivalent form of the game given in Figure 5.7a due to the tree structure of the game and the fact that there are more than one connection to the same point (for example, there are connections to S7 from both S2 and S3) and the strategy determination process of the game starts by considering this new tree structure depicted in Figure 5.7b within the context of Remark 5.1.2.

Let the transition matrix of the game $P \in \mathbb{R}^{3\times 3}$ be given as

$$P = \begin{bmatrix} U & 0.6 & 0.2 & 0.2 \\ S & 0.15 & 0.70 & 0.15 \\ D & 0.05 & 0.15 & 0.8 \end{bmatrix}$$

where U, S, and D stand for Up, Straight, and Down actions respectively.

To solve the game, the derivation of matrices A_s begins from the final stage, that is the 3^{rd} stage for this game.



(a) (b)

Figure 5.7 : The original game tree on the left and the equivalent game tree on the right.

For the Markov reward game depicted in Figure 5.7b, which comprises 3 stages, the matrix $A_3 \in \mathbb{R}^{3\times 7}$ is derived by considering the rewards in the 3rd stage from up to down according to Step 1 as displayed in Figure 5.2.

$$A_3 = \begin{bmatrix} 9 & 0 & 2 & 5 & 7 & 1 & 12 \\ 7 & 6 & 8 & 11 & 3 & 10 & 6 \\ 0 & 2 & 5 & 7 & 1 & 12 & 6 \end{bmatrix}.$$

Then by Step 2, the moving matrix $M_3^c \in \mathbb{R}^{3\times 7}$ is obtained as $M_3^c = PA_3$,

$$M_3^c = \begin{bmatrix} 6.8 & 1.6 & 3.8 & 6.6 & 5 & 5 & 9.6 \\ 6.25 & 4.5 & 6.65 & 9.5 & 3.3 & 8.95 & 6.9 \\ 1.5 & 2.5 & 5.3 & 7.5 & 1.6 & 11.15 & 6.3 \end{bmatrix}.$$

Following Step 3, ∞-norms are evaluated by considering each column of the moving matrix M_3^c and $S_3(:,j) = ||M_3^c(:,j)||_\infty$, $j = 1,2,...,7$ is generated as,

$$S_3 = \begin{bmatrix} 6.8 & 4.5 & 6.65 & 9.5 & 5 & 11.15 & 9.6 \\ 6.8 & 4.5 & 6.65 & 9.5 & 5 & 11.15 & 9.6 \\ 6.8 & 4.5 & 6.65 & 9.5 & 5 & 11.15 & 9.6 \end{bmatrix}.$$

Continuing to Step 4 and as a final step for the 3rd stage, we generate the matrix providing the optimal strategy for the stage St_3 by $St_3 = M_3^c - S_3$ as,

$$St_3 = \begin{bmatrix} 0 & -2.9 & -2.85 & -2.9 & 0 & -6.15 & 0 \\ -0.55 & 0 & 0 & 0 & -1.7 & -2.2 & -2.7 \\ -5.3 & -2 & -1.35 & -2 & -3.4 & 0 & -3.3 \end{bmatrix}.$$

The zeros in each column represent the optimal strategy for the related stages and the optimal strategy for the 3rd stage is {U, S, S, S, U, D, U} derived from the places of the zeros in the matrix St_3.

Then, we start solving the game for the 2^{nd} stage similarly. By Step 1, we first create the matrix $A_2 \in \mathbb{R}^{3\times 3}$ representing the rewards in the corresponding stage as,

$$A_2 = \begin{bmatrix} 4 & 8 & 3 \\ 9 & 11 & 10 \\ 8 & 3 & 7 \end{bmatrix}.$$

As an intermediate step, which is expressed in Steps 5 and 6 implying the update process of the matrix A_2. Let $t = 0$, for each $j = 1,2,3$ corresponding to all $i = 1,2,3$, then $t = t+1$ and $A_2^U(i,j) = ||M_3^c(:,t)||_1 + A_2(i,j)$ is derived as,

$$A_2^U = \begin{bmatrix} 18.55 & 23.75 & 12.9 \\ 17.6 & 34.6 & 35.1 \\ 23.75 & 12.9 & 29.8 \end{bmatrix},$$

then we redefine $A_2 = A_2^U$. As in As in Step 2, by using $M_2^c = PA_2$, the following moving matrix $M_2^c \in \mathbb{R}^{3\times 3}$ is obtained,

$$M_2^c = \begin{bmatrix} 19.4 & 23.75 & 20.72 \\ 18.665 & 29.7175 & 30.975 \\ 22.5675 & 16.6975 & 29.75 \end{bmatrix}.$$

Following Step 3, we create the matrix $S_2(:,j) = ||M_2^c(:,j)||_\infty$, $j = 1,2,3$ by considering each column of the moving matrix M_2^c as,

$$S_2 = \begin{bmatrix} 22.5675 & 29.7175 & 30.975 \\ 22.5675 & 29.7175 & 30.975 \\ 22.5675 & 29.7175 & 30.975 \end{bmatrix}.$$

As the last step, i.e. Step 4, the matrix St_2 giving the optimal strategy for the 2^{nd} stage is obtained as {D, S, S}, where

$$St_2 = \begin{bmatrix} -3.1675 & -5.9675 & -10.255 \\ -3.9025 & 0 & 0 \\ 0 & -13.02 & -1.225 \end{bmatrix}.$$

For the determination of the optimal strategy in the 1^{st} stage, the matrix $A_1 \in \mathbb{R}^{3\times 1}$ is generated as in Step 1

$$A_1 = \begin{bmatrix} 7 \\ 4 \\ 8 \end{bmatrix}.$$

The matrix A_1 is updated as an intermediate step implied in Steps 5 and 6. Let $t = 0$, for each $j = 1$ corresponding to all $i = 1,2,3$, then $t = t+1$ and $A_1^U(i,j) = ||M_2^c(:,t)||_1 + A_1(i,j)$ is

$$A_1^U = \begin{bmatrix} 67.6325 \\ 74.165 \\ 89.445 \end{bmatrix}.$$

Then, $A_1 = A_1^U$ is redefined and create the moving matrix $M_1^c \in \mathbb{R}^{3\times 1}$ by $M_1^c = PA_1$ according to Step 2 as

$$M_1^c = \begin{bmatrix} 73.3015 \\ 75.477125 \\ 86.062375 \end{bmatrix}.$$

Next by Step 3, $S_1(:,j) = ||M_1^c(:,j)||_\infty$, $j = 1$ is evaluate by considering the column of the moving matrix M_1^c as

$$S_1 = \begin{bmatrix} 86.062375 \\ 86.062375 \\ 86.062375 \end{bmatrix}.$$

As in Step 4, the matrix giving the optimal strategy for 1st stage $St_1 = M_1^c - S_1$ is obtained as {D}, where

$$St_1 = \begin{bmatrix} -12.760875 \\ -10.58525 \\ 0 \end{bmatrix}.$$

For the initial stage, we only apply Step 5 of Algorithm 1 to update its reward as

$$A_0 = [0]\ for\ A_0^U = ||M_1^c(:,t)||_1 + A_0(i,j),\ i,j = 1.$$

$$A_0^U = [234.841] = A_0.$$

Thus the computation process is completed. Finally, the optimal strategy for the Markov reward game is determined by considering the matrices St_1, St_2, St_3. By compiling each optimal path associated with the respective stage, the optimal path is determined as $\{D,D,S,S,U,S,S,S,U,D,U\}$ by Step 6.

5.3 Markov Reward Games with Artificial Intelligence

Artificial Neural Networks and Convolutional Neural Networks have advanced the fields of machine learning and artificial intelligence. These techniques enable computers to learn from data and perform complex tasks related to classification, clustering, or prediction with notably high accuracy [120–122]. Convolutional Neural Networks are a specific type of artificial neural network primarily utilized for image processing but also applicable to predictive tasks [123, 124]. CNNs consist of convolutional layers that apply filters to the input data and use pooling layers to reduce its dimensions. This enables the CNN to learn and extract important features from the data. These processes allow the convolutional network to identify patterns and relationships within the data that are crucial for addressing complex problems [76].

In this section, we aim to utilize Convolutional Neural Networks (CNNs) to address Markov reward games represented as decision trees, as illustrated in Figure 5.1. Traditionally, these games are tackled using iterative methods such as value iteration, policy iteration, and dynamic programming. These methods involve updating processes for each state, resulting in increased computational complexity as the number of stages or actions increases, as emphasized in Section 1.1.4 of Chapter 1. To address this complexity, we adopt the idea of transforming the decision tree into a payoff matrix, as presented in Figure 5.2 in Section 5.1.1. This transformation allows us to effectively feed the system with the necessary input for the proposed convolutional neural network. For simplicity, we consider Markov reward games with three states and three actions, as illustrated in Example 2 of Section 5.2.

First, we generated two different datasets containing 3000 and 5000 Markov reward games with three stages and three actions for the training and testing processes of the neural network proposed in Figure 5.8. Next, we obtained the optimal solutions for each game using the matrix norm-based method introduced in Section 5.1.1. We reserved 80% of the dataset for training the neural network and used 20% for testing the model's performance on unseen data. Figure 5.8 illustrates the general structure of the proposed neural network. The decision tree in Figure 5.8 is transformed into an appropriate payoff matrix, as shown in Figure 5.2. We present the details of the neural

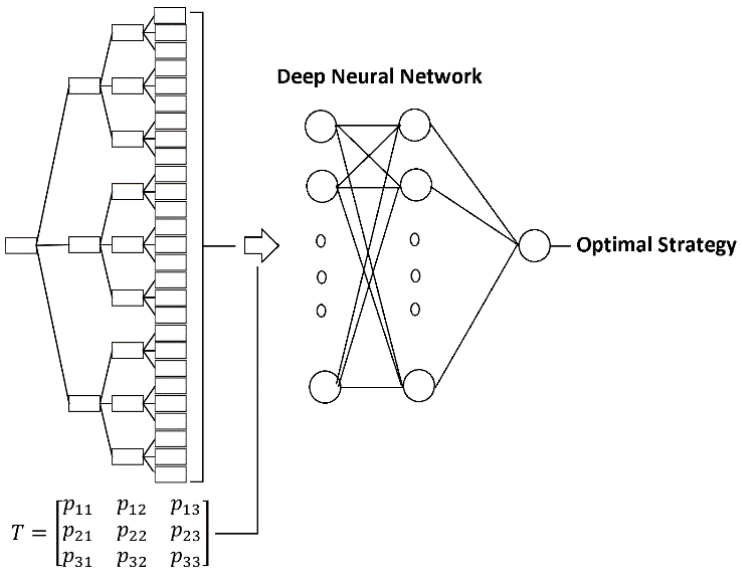


Figure 5.8 : Convolutional neural network architecture for prediction of optimal strategies for Markov reward games with 3-stages and 3-actions

network architecture as follows [125]:

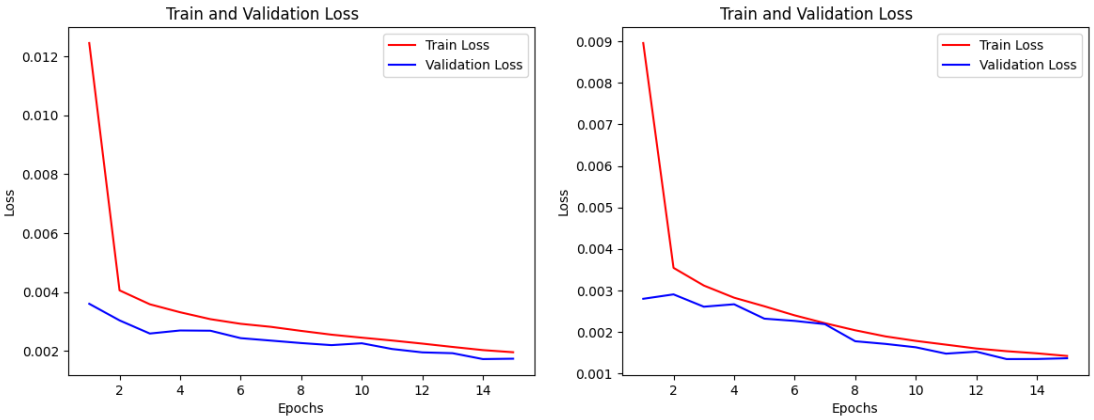
- Firstly, the rewards in the game are normalized to have zero mean and unit variance for the training dataset. 3 different convolutional layers are applied on the data, where each layer has 3×1 filters and Linear activation function. The number of filters in the layers are 256, 128, 64 respectively.

- Then, MaxPooling is implemented between the convolutional layers. Dropout is added to Layer 1 and Layer 3 to prevent the learning process from overfitting. Next, a Flatten operation is applied to convert the output into a one-dimensional vector.

- Following that, the output is sent to 2 Dense layer with output sizes 128 and 64, 39 with Linear activation. Finally, a reshape operation is applied.

We used RMSprop as an optimizer, MSE as a loss function and MAE as a metrics. We trained the model for 1000 epochs but learning ended when the validation loss did not improve for 15 epochs.

Figures 5.9a and 5.9b illustrate how the loss error in both validation and training data varies with different dataset sizes, respectively 3000 and 5000 games, respectively. The data clearly indicate that error rates decrease consistently as the sample size increases, that reveals a positive relationship between the size of dataset and efficiency of learning



(a) Training and Validation Loss for 3000 Markov Games

(b) Training and Validation Loss for 5000 Markov Games

Figure 5.9 : Changes in the training loss and validation loss

process. This result is typical in deep learning process, that more pattern among the inputs ant outputs obtained by more data.

Furthermore, Figure 5.9 demonstrates that predictions for optimal rewards improve with larger dataset sizes. This enhancement implies that the model gets better at extrapolating from the training data to new data, which is essential for real-world uses. In this context, the performance of the trained model on test data yields mean absolute errors of 0.0327 for the 3000 games dataset and 0.0284 for the 5000 games dataset. As anticipated, a larger sample size is necessary to achieve a lower error rate, highlighting that the rate of error decay is influenced by the sample size.

Consequently, our findings highlight how crucial dataset size is for neural network training and recommend that future research concentrate on growing the dataset even more in order to investigate the boundaries of model performance. We may be able to attain even lower error rates and more accurate predictions by continuously expanding the dataset size and improving the neural network's architecture, which would ultimately increase CNNs' suitability in challenging decision-making situations.

6. CONCLUSIONS AND DISCUSSIONS

We extend the approach used for solving zero-sum matrix games through matrix norms to nonzero-sum bimatrix games by introducing new notation in Chapter 2. We present and prove theorems that establish inequalities only depending on the 1-norm and ∞-norm of each player's payoff matrix, focusing on the game value in bimatrix scenarios. Additionally, we delineate the lower and upper bounds for the maximum and minimum elements within the strategy set. Building on existing literature, we enhance the boundaries for game values in both zero-sum and nonzero-sum matrix games, resulting in an improved interval for the game value. We also reformulate the min-max theorem specifically for bimatrix games. A key advantage of our approach is that it allows for faster approximation of solutions for nonzero-sum bimatrix games, as the MN method bypasses linear programming and other computational techniques, providing results without the need to solve any systems of equations. This aspect significantly strengthens the application of our methodology. Furthermore, we conduct a thorough analysis of the convergence of both the MN method and the Extended Matrix Norm method, presenting our findings as a theorem, along with a corresponding corollary that demonstrates the existence of a solution within the matrix norm methods's framework.

We demonstrate the extended matrix norm method by applying it to the classic "Battle of the Sexes" problem and other bimatrix games. Our findings confirm that this approach is consistent with existing literature and highlight significant advancements in the boundary estimations for game values. We comprehensively demonstrate the convergence results of the EMN method using some of these examples. In addition to these, we present a game theoretical model using repeated bimatrix games for the risk of infection during Covid-19 pandemic. We run our analysis on Türkiye, South Korea and Italy and show their quarantine behaviours during the health crisis. We conclude that keeping the quarantine rules decreases the risk of infection.

In Chapter 3, we focus on eliminating the challenge in distributing the elements of the strategy set while applying the EMN method, especially for large-scale zero-sum (LS-ZS) matris games. In this context, we aim to support the EMN method for boosting its performance and overcome the distribution problem. For this purpose, we provide the algorithm for the creation process for any size of matrix games and generate matrix games in various sizes and numbers to create a dataset. Then, we establish a machine learning model that can leverage the EMN method to obtain approximate values for LS-ZS matrix games. Our model consists of a neural network that is trained on a large number of random zero-sum matrix games, that is the dataset contains 500-50000 matrix games, and approximate solutions offered by the EMN algorithm. We demonstrate that the proposed machine learning model can obtain high-quality approximate value prediction for LS-ZS matrix games. We compare the results obtained for 5×5, 10×10, and 50×50 by tables and graphs. The results suggest that training the neural network with larger datasets can lead to improved approximations of game values, as generally expected. A key advantage of our methodology is that, following the training phase, solutions to matrix games can be derived in constant time through neural network inference. This efficiency makes our approach suitable for real-world problems involving large matrices.

In Chapter 4, we begin by exploring stochastic matrix games and the widely-used Shapley iteration as a solution method for these games. We then introduce four alternatives to Shapley iteration for solving stochastic matrix games. First, we combine the EMN method with Shapley iteration to create a novel approach called the weak and strong hybrid Shapley iterations. In weak hybrid Shapley iteration, the EMN acts as an intermediary step, facilitating faster convergence to the game value compared to traditional Shapley iteration. Next, we enhance this by integrating the iterative EMN (IEMN) with Shapley iteration, leading to a more robust method termed the strong hybrid Shapley iteration. This stronger approach yields a more accurate approximation of the game value than the weak hybrid method, significantly improving convergence. We then proposed two algorithms that based on the EMN and IEMN methods, eliminating the need for analytical solutions to stochastic matrix games. Additionally, we introduced the semi-extended matrix norm and semi-iterative extended matrix norm methods, which leverage the direct outputs of the EMN and IEMN methods without

requiring the establishment of a specific strategy set, thereby simplifying the approach. While semi-extended matrix norm and semi-iterative extended matrix norm methods demonstrate satisfactory performance, their efficiency improves when a proper strategy set is created using the relevant theorem. Notably, the IEMN method outperforms the EMN method due to its iterative advantages.

We applied our proposed methods to stochastic matrix games of dimensions 2×2 and 3×3, computing and comparing their relative errors. The results showed that the strong hybrid Shapley iteration significantly outperforms both the Shapley iteration and all other proposed methods. Furthermore, both hybrid methods notably exceed the performance of the extended matrix norm and iterative extended matrix norm methods. To evaluate scalability, we generated a dataset of 1000 stochastic matrix games of sizes 5×5 and 10×10. The performance ranking revealed that the strong hybrid Shapley iteration method leads, followed by the weak hybrid Shapley iteration, semi-iterative extended matrix norm, and semi-extended matrix norm methods. These findings demonstrate that our proposed methods achieve comparable performance to the Shapley iteration with lower computational costs. To the best of our knowledge, the extended matrix norm and iterative extended matrix norm methods are pioneering approaches for solving stochastic matrix games by leveraging matrix norms.

In Chapter 5, we integrated matrix norms into the solution process for single-agent stochastic games, particularly focusing on Markov reward games represented as decision trees. Contrary to the traditional methods such as value iteration, policy iteration, and dynamic programming, our matrix norm-based approach addresses the game holistically rather than analyzing each state and its actions separately. This new method processes stages and corresponding actions simultaneously, calculating rewards collectively for all actions. To implement this approach, we propose a method to convert the game tree into an appropriate matrix format. We introduced the concept of the moving matrix, which facilitates the integration of stage rewards and the transition matrix. Additionally, we presented a strategy matrix that indicates the optimal strategy for each stage, using zeros for clarity in results for practitioners.

One of the key advantages of the matrix norm-based method is that it eliminates the need for repeated computations to determine the optimal strategy for a stage, unlike value iteration or policy iteration, which can prolong the process. We then provide

an algorithm and a diagram illustrating the application of this method. Although the approach initially seemed applicable only to games with positive rewards, we address this limitation by perturbing the rewards while maintaining an invariant strategy set, as discussed in Remark 5.1.1. Additionally, we establish a connection between actual and perturbed rewards in Remark 5.1.3, allowing for the retrieval of actual rewards using the provided equation. Furthermore, we introduce a modification technique that involves creating a dual tree of the original game, applicable when multiple nodes connect to the same node, which we elaborate on in Remark 5.1.2.

Alongside establishing the new alternative method, we illustrate its application using a solution diagram for a benchmark Markov reward game having 2-actions and 2-stages. This example also demonstrates how the method can be applied to Markov reward games with negative rewards, explicitly utilizing Remark 5.1.3. Additionally, we present a comprehensive application of the matrix norm-based method through a solution algorithm for a larger Markov reward game involving 3-actions and 3-stages. This example further explains the application of the method when multiple nodes connect to the same node in the game tree. Both examples confirm that the decision trees can be converted into matrix form by following the algorithm and applying Remarks 5.1.1 and 5.1.2, depending on the rewards and node connections. Consequently, we have shown that the matrix norm-based method is effective for tree-structured games, particularly Markov reward games. Moreover, this method serves as a practical alternative and a complement to existing literature, incorporating innovations such as the moving matrix concept, matrix norms, and a holistic approach to action consideration.

Additionally, we integrate the proposed method with machine learning by developing a convolutional neural network architecture for solving these games. To train the neural network, we generate two datasets consisting of 3,000 and 5,000 Markov reward games with randomly generated rewards, allowing us to compare the results across these datasets. The neural network uses the payoff matrix obtained by proposed method and the transition matrix as an input and provide optimal strategies as an output. The findings confirm that larger datasets enhance model accuracy, as generally anticipated. A significant contribution of our work is that, once trained for a specific problem size, the solution time remains constant. This contrasts with existing solution methods,

which require ongoing updates and problem-specific modeling. For future research, we aim to enhance the proposed model to tackle more complex games.

REFERENCES

[1] **İzgi, B. and Özkaya, M.** (2019). A New Perspective to the Solution and Creation of Zero Sum Matrix Game with Matrix Norms, *Applied Mathematics and Computation*, *341*, 148–159.

[2] **Borel, E.** (1921). La théorie du jeu et les équations intégrales à noyau symétrique, *Comptes Rendus de l'Académie des Sciences*, *173*, 1304–1308.

[3] **Borel, E.** (1924). Sur les jeux où interviennent l'hasard et l'habileté des joueurs, *Theorie des Probabilités, Paris: Librairie Scientifique*, *173*, 204–224.

[4] **Borel, E.** (1924). Un théorème sur les systèmes de formes linéaires à déterminant symétrique gauche, *Comptes Rendus de l'Académie des Sciences*, *173*, 925–927.

[5] **Borel, E.** (1928). Sur les systèmes de formes linéaires à déterminant symétrique gauche et la théorie générale du jeu, *Comptes Rendus de l'Académie des Sciences*, *173*, 52–53.

[6] **von Neumann, J.** (1928). Zur theorie der gesellschaftsspiele, *Mathematische annalen*, *100*(1), 295–320.

[7] **von Neumann, J. and Morgenstern, O.** (1944). *Theory of Games and Economic Behavior*, Princeton University Press.

[8] **Nash, J.F.** (1950). Equilibrium Points in n-Person Games, *Proceedings of the National Academy of Sciences*, *36*(1), 48–49.

[9] **Nash, J.F.** (1950). The Bargaining Problem, *Econometrica*, *18*(2), 155–162.

[10] **Shapley, L.S.** (1953). A Value for n-Person Games, *Contributions to the Theory of Games (AM-28)*, *2*, 307–317.

[11] **Shapley, L.S.** (1953). Stochastic Games, *Proceedings of the National Academy of Sciences*, *39*(10), 1095–1100.

[12] **Shapley, L.S. and Shubik, M.** (1954). A Method for Evaluating the Distribution of Power in a Committee System, *American Political Science Review*, *48*(3), 787–792.

[13] **Gillies, D.B.** (1959). Solutions to General Non-Zero-Sum Games, *Contributions to the Theory of Games (AM-40)*, *4*, 47–86.

[14] **Schelling, T.C.** (1960). *The Strategy of Conflict*, Harvard University Press.

[15] **Rufus, I.** (1965). *Differential Games: A Mathematical Theory with Applications to Warfare and Pursuit, Control and Optimization*, John Wiley and Sons.

[16] **Harsanyi, J.C.** (1967). Games with Incomplete Information Played by "Bayesian" Players, I–III Part I. The Basic Model, *Management Science*, *14*(3), 159–182.

[17] **Shapley, L.S.** (1967). On Balanced Sets and Cores, *Naval Research Logistics Quarterly*, *14*(4), 453–460.

[18] **Lucas, W.F.** (1968). A Game with No Solution, *Bulletin of the American Mathematical Society*, *74*(2), 237–240.

[19] **Maynard Smith, J. and Price, G.** (1973). The Logic of Animal Conflict, *Nature*, *264*, 15–18.

[20] **Selten, R.** (1975). Reexamination of the Perfectness Concept for Equilibrium Points in Extensive Games, *International Journal of Game Theory*, *4*(1), 25–55.

[21] **Conway, J.H.** (1976). *On Numbers and Games*, Academic Press.

[22] **Berlekamp, E.R.**, **Conway, J.H. and Guy, R.K.** (1982). *Winning Ways for Your Mathematical Plays*, Academic Press.

[23] **Harsanyi, J.C. and Selten, R.** (1988). *A General Theory of Equilibrium Selection in Games*, The MIT Press.

[24] **Roth, A.E. and Sotomayor, M.** (1992). Two-sided Matching, *Handbook of Game Theory with Economic Applications*, (16), 485–541.

[25] **Aumann, R.J. and Maschler, M.B.** (1995). *Repeated Games with Incomplete Information*, The MIT Press.

[26] **Milgrom, P.R.** (2004). *Putting Auction Theory to Work*, Cambridge University Press.

[27] **Nisan, N.**, **Roughgarden, T.**, **Tardos, E. and Vazirani, V.V.** (2007). *Algorithmic Game Theory*, Cambridge University Press.

[28] **Camerer, C.F.** (2011). *Behavioral Game Theory: Experiments in Strategic Interaction*, Princeton University Press.

[29] **Sánchez-Pérez, J.** (1992). A New Look at the Study of Solutions for Games in Partition Function Form, *In Recent Advances in Game Theory and Applications: European Meeting on Game Theory*, (16), 225–249.

[30] **Clempner, J.B.** (2018). On Lyapunov Game Theory Equilibrium: Static and Dynamic Approaches, *International Game Theory Review*, *20*(02), 1750033.

[31] **Blanc, P. and Rossi, J.D.** (2019). *Game Theory and Partial Differential Equation*, Walter de Gruyter GmbH & Co KG.

[32] **Deng, X.**, **Jiang, W. and Wang, Z.** (2019). Zero-sum Polymatrix Games with Link Uncertainty: A Dempster-Shafer Theory Solution, *Applied Mathematics and Computation*, *340*, 101–112.

[33] **Pangallo, M.**, **Heinrich, T. and Doyne Farmer, J.** (2019). Best Reply Structure and Equilibrium Convergence in Generic Games, *Science Advances*, *5*(2), eaat1328.

[34] **Liu, H.** (2020). The Relation Between Eigenvalue/Eigenvector and Matrix Game, *arXiv preprint*, arXiv:2012.01269.

[35] **Babajanyan, S.G.**, **Allahverdyan, A.E. and Cheong, K.H.** (2020). Energy and Entropy: Path from Game Theory to Statistical Mechanics, *Physical Review Research*, *2*(4), 043055.

[36] **Malhotra, T.** (2021). *Computation with 2-Tuple Linguistic Variables and Its Application in Matrix Games*, Ph.D. Thesis, Delhi Technological University.

[37] **Yekkehkhany, A.**, **Murray, T. and Nagi, R.** (2021). Stochastic Superiority Equilibrium in Game Theory, *Decision Analysis*, *18*(2), 153–168.

[38] **Petrosyan, L.A. and Yeung, D.W.K.** (2021). Shapley Value for Differential Network Games: Theory and Application, *The Journal of Dynamics and Game*, *8*(2), 151–166.

[39] **Zheng, Y. and Shi, J.** (2022). Stackelberg Stochastic Differential Game with Asymmetric Noisy Observations, *International Journal of Control*, *95*(9), 2510–2530.

[40] **Hazra, T. and Anjaria, K.** (2022). Applications of Game Theory in Deep Learning: A Survey, *Multimedia Tools and Applications*, *81*(6), 8963–8994.

[41] **Broom, M. and Rychtář, J.** (2022). *Game-Theoretical Models in Biology*, Chapman and Hall/CRC.

[42] **İzgi, B.**, **Özkaya, M.**, **Üre, N.K. and Perc, M.** (2023). Extended Matrix Norm Method: Applications to Bimatrix Games and Convergence Results, *Applied Mathematics and Computation*, *438*, 127553.

[43] **İzgi, B.**, **Özkaya, M.**, **Üre, N.K. and Perc, M.** (2023). Machine Learning Driven Extended Matrix Norm Method for the Solution of Large-Scale Zero-Sum Matrix Games, *Journal of Computational Science*, *68*, 101997.

[44] **Djebara, S.**, **Achemine, F. and Zerdani, O.** (2023). A New Approach for Solving Constrained Matrix Games with Fuzzy Constraints and Fuzzy Payoffs, *Journal of Mathematical Modeling*, *11*(3), 425–439.

[45] **Djebara, S.**, **Achemine, F. and Zerdani, O.** (2023). Stochastic Adaptive Linear Quadratic Differential Games, *IEEE Transactions on Automatic Control*, *69*(2), 1066–1073.

[46] **Lozovanu, D. and Pickl, S.W.** (2024). *Markov Decision Processes and Stochastic Positional Games*, Springer Cham.

[47] **İzgi, B.**, **Kocken, H.G. and Özkaya, M.** (2024). Matrix Norm Methods for Zero-Sum Fuzzy Matrix Games with Payoffs of Triangular Fuzzy Numbers, *Applied Mathematics and Computation*, *479*(2), 128874.

[48] **Shreyas, S.R. and Vijesh, A.** (2024). A Two-Step Minimax Q-learning Algorithm for Two-Player Zero-Sum Markov Games, *arXiv preprint*, arXiv:2407.04240.

[49] **İzgi, B.**, **Özkaya, M.**, **Üre, N.K. and Perc, M.** (2024). Matrix Norm Based Hybrid Shapley and Iterative Methods for the Solution of Stochastic Matrix Games, *Applied Mathematics and Computation*, *473*, 128639.

[50] **Crönert, T. and Minner, S.** (2024). Equilibrium Identification and Selection in Finite Games, *Operations Research*, *72*(2), 816–831.

[51] **Dey, I.**, **Marchetti, N.**, **Caleffi, M. and Cacciapuoti, A.S.** (2024). Quantum Game Theory Meets Quantum Networks, *IEEE Wireless Communications*, *31*(4), 90–96.

[52] **İzgi, B.**, **Özkaya, M.**, **Üre, N.K. and Perc, M.** (2025). A Holistic Matrix Norm-Based Alternative Solution Method for Markov Reward Games, *Applied Mathematics and Computation*, *488*, 129124.

[53] **Ferguson, T.S.** (2014). *Game Theory Part III*, Mathematics Department UCLA.

[54] **Brown, K.L. and Shoham, Y.** (2008). *Essentials of Game Theory*, Morgan & Claypool, 1st edition.

[55] **Prisner, E.** (2014). *Game Theory Through Examples*, The Mathematical Association of America.

[56] **Dresher, M.** (1961). *Games of Strategy: Theory and Application*, Prentice-Hall.

[57] **Özkaya, M.** (2018). *The Roles of Matrix Norms in the Game Theory*.

[58] **Meyer, C.D.** (2023). *Matrix Analysis and Applied Linear Algebra*, Society for Industrial and Applied Mathematics (SIAM).

[59] **Solan, E. and Vieille, N.** (2015). Stochastic Games, *Proceedings of the National Academy of Sciences*, *112*(45), 13743–13746.

[60] **Friedman, A.** (1972). Stochastic Differential Games, *Journal of Differential Equations*, *11*(1), 79–108.

[61] **Rao, A. and Jelvis, T.** (2022). *Foundations of Reinforcement Learning with Applications in Finance*, Chapman and Hall/CRC.

[62] **Howard, R.A.** (1960). *Dynamic Programming and Markov Processes*, MIT Press & John Wiley and Sons Inc.

[63] **Sutton, R.S. and G., B.A.** (2018). Reinforcement learning: An introduction.

[64] **Büyüktahtakin, I.E.** (2011). Dynamic Programming via Linear Programming.

[65] **Hillier, F.S. and Lieberman, G.J.** (1995). Introduction to Mathematical Programming, *Journal of Mathematics and Mechanics*.

[66] **Duran, A. and İzgi, B.** (2015). Application of the Heston Stochastic Volatility Model for Borsa Istanbul Using Impression Matrix Norm, *Journal of Computational and Applied Mathematics*, *281*, 126–134.

[67] **İzgi, B.** (2015). Behavioral Classification of Stochastic Differential Equations in Mathematical Finance, *Ph.D. Thesis, Istanbul Technical University, Istanbul, Türkiye*.

[68] **McCarthy, J.**, **Minsky, M.L.**, **Rochester, N. and Shannon, C.A.** (1955). Proposal for the Dartmouth Summer Research Project on Artificial Intelligence, *AI Magazine*, *27*(4), 12–14.

[69] **Findler, N.V.** (1988). The Debt of Artificial Intelligence to John von Neumann, *Artificial Intelligence Review*, *2*(3), 311–312.

[70] **Simon, H.** (1981). *The Sciences of the Artificial*, The MIT Press, Cambridge, MA.

[71] **Tennenholtz, M.**, (2002). Game theory and artificial intelligence, Foundations and applications of multiagent systems, Springer, Berlin, p.49–58.

[72] **Elkind, E. and Leyton-Brown, K.** (2010). Algorithmic game theory and artificial intelligence, *AI Magazine*, *31*(4), 9–12.

[73] **Hazra, T. and Anjaria, K.** (2022). Applications of game theory in deep learning: a survey, *Multimedia Tools and Applications*, *81*(6), 8963–8994.

[74] **Zhou, Y.**, **Kantarcioglu, M. and Xi, B.** (2019). A survey of game theoretic approach for adversarial machine learning, *Wiley Interdiscip Rev Data Min Knowl Disco.*, *9*(3), e1259.

[75] **Lippi, M.** (2015). Statistical relational learning for game theory, *IEEE Trans Comput Intell AI Games*, *8*(4), 412–425.

[76] **Chollet, F.** (2018). *Deep Learning with Python*, Manning Publications Co., New York, USA.

[77] **Wu, Y.C. and Feng, J.W.** (2018). Development and application of artificial neural network, *Wireless Personal Communications*, *102*, 1645–1656.

[78] **Oliveira, R.M.D.**, **Araújo, R.C.**, **Barros, F.J.**, **Segundo, A.P.**, **Zampolo, R.F.**, **Fonseca, W. and Brasil, F.S.** (2017). A system based on artificial neural networks for automatic classification of hydro-generator stator windings partial discharges, *Journal of Microwaves, Optoelectronics and Electromagnetic Applications*, *16*(03), 628–645.

[79] **DeepAi** (2023). *Weights - Artificial Neural Networks*, `https://deepai.org/machine-learning-glossary-and-terms/weight-artificial-neural-network`, accessed: 19.12.2023.

[80] **Turing** (2023). *What is the necessity of bias in neural networks?*, https://www.turing.com/kb/necessity-of-bias-in-neural-networks#why-is-bias-added-in-neural-networks, accessed: 15.12.2023.

[81] **Joshi, N.** (2019). *3 types of neural network that AI uses*, https://www.allerin.com/blog/3-types-of-neural-networks-that-ai-uses, accessed: 26.12.2023.

[82] **Zvornicanin, E.** (2023). *Convolutional neural network vs. regular neural network*, https://www.baeldung.com/cs/convolutional-vs-regular-nn, accessed: 26.12.2023.

[83] **IBM** (n.d.). *What are convolutional networks?*, https://www.ibm.com/topics/convolutional-neural-networks, accessed: 26.12.2023.

[84] **Mueller, P. and Massaron, L.** (2021). *Machine Learning for Dummies*, John Wiley & Sons, Inc., 2 edition.

[85] **Narayan, S.** (1997). The generalized sigmoid activation function: Competitive supervised learning, *Information Sciences*, *99*(1-2), 69–82.

[86] **Agarap, A.F.** (2018). Deep learning using rectified linear units (ReLU), *arXiv*, arXiv preprint arXiv:1803.08375.

[87] **Dubey, S.R.**, **Singh, S.K. and Chaudhuri, B.B.** (2022). Activation functions in deep learning: A comprehensive survey and benchmark, *Neurocomputing*, *503*, 92–108.

[88] **Kong, S. and Takatsuka, M.** (2021). Hexpo: A vanishing-proof activation function, ***Y. Choe**, editor, International Joint Conference on Neural Networks*, pp.2562–2567.

[89] **Misra, D.** (2019). Mish: A self-regularized non-monotonic activation function, *arXiv preprint*, arXiv:1908.08681.

[90] **Kandel, I. and Castelli, M.** (2020). Transfer learning with convolutional neural networks for diabetic retinopathy image classification: A review, *Applied Sciences*, *10*(6), 2021.

[91] **Ruder, S.** (2016). An overview of gradient descent optimization algorithms, *arXiv preprint*, arXiv:1609.04747.

[92] **Starmer, J.** (2022). *The Statquest illustrated guide to machine learning!!!: master the concepts, one full-color picture at a time, from the basics all the way to neural networks*, BAM!

[93] **Tuyls, K. and Nowé, A.** Evolutionary game theory and multi-agent reinforcement learning, *Knowl. Eng. Rev.*, *20*(1), 63–90.

[94] **Schuster, A. and Yamaguchi, Y.**, (2010). Application of game theory to neuronal networks, Advances in Artificial Intelligence, p.521606.

[95] **Agrawal, A. and Jaiswal, D.** (2012). When machine learning meets AI and game theory, *Computer Science*.

[96] **Madani, K. and Hooshyar, M.** (2014). A game theory–reinforcement learning (GT–RL) method to develop optimal operation policies for multi-operator reservoir systems, *J. Hydrol.*, *519*(A), 732–742.

[97] **Xiao, L.**, **Li, Q.**, **Chen, T.**, **Cheng, E. and Dai, H.** (2015). Jamming Games in Underwater Sensor Networks with Reinforcement Learning, *IEEE Global Communications Conference*, 1–6.

[98] **Rekha, U.J.**, **Shahu, C.K. and Vinaya, B.A.** (2016). Game theory and its applications in machine learning, *Information Systems Design and Intelligent Applications: Proceedings of Third International Conference INDIA*, *3*, 195–207.

[99] **Stier, J.**, **Gianini, G.**, **Granitzer, M. and Ziegler, K.** (2018). Analysing neural network topologies: a game theoretic approach, *Procedia Computer Science*, *126*, 234–243.

[100] **Moran, N. and Pollack, J.**, (2018). Coevolutionary neural population models, In Artificial Life, p.39–46.

[101] **Albaba, B.M. and Yıldız, Y.** (2021). Driver modeling through deep reinforcement learning and behavioral game theory, *IEEE Trans. Control Syst. Technol.*, *30*(2), 885–892.

[102] **Wu, D. and Lisser, A.** (2022). A dynamical neural network approach for solving stochastic two-player zero-sum games, *Neural Networks*, *152*, 140–149.

[103] **Khabbouchi, I.**, **Said, D.**, **Oukaira, A.**, **Mellal, I. and Khoukhi, L.** (2023). Machine Learning and Game-Theoretic Model for Advanced Wind Energy Management Protocol (AWEMP), *Energies*, *16*(5), 2179.

[104] **Guseinov, K.G.**, **Akyar, E. and Duzce, S.A.** (2010). *Oyun Teorisi Çatışma ve Anlaşmanın Matematiksel Modelleri*, Seçkin.

[105] **İzgi, B. and Özkaya, M.** (2019). Demonstration of the Fairness of a Matrix Game with Matrix Norms, *Int. J. Adv. Eng. Pure. Sci*, *31*(2), 126–132.

[106] **İzgi, B. and Özkaya, M.** (2021). The Demonstration of the Necessity of Agriculture Insurance by Game Theory: Matrix Norm Approach, *AKU J. Sci. Eng.*, *20*, 824–831.

[107] **Barron, N.E.** (2013). *Game Theory: An Introduction*, John Wiles & Sons Inc.

[108] **Özkaya, M. and İzgi, B.** (2021). Effects of the Quarantine on the Individuals' Risk of Covid-19 Infection: Game Theoretical Approach, *Alexandria Engineering Journal*, *60*(4), 4157–4165.

[109] **Our World in Data** (2021). https://ourworldindata.org/covid-testing, accessed: 08.01.2021.

[110] **John Hopkins University Coronovirus** (2021). https://coronavirus.jhu.edu/map.html, accessed: 08.01.2021.

[111] **Straffin, P.** (1993). *Game Theory and Strategy*, The Mathematical Association of America.

[112] **Digital, G.** (2021). *Coronavirus: I contagi in Italia*, https://lab.gedidigital.it/gedi-visual/2020/coronavirus-i-contagi-in-italia/, accessed: 08.01.2021.

[113] **Türkiye Cumhuriyeti Sağlık Bakanlığı** (2021). *Covid-19 Verileri*, https://covid19.saglik.gov.tr, accessed: 08.01.2021.

[114] **Özkaya, M. and İzgi, B.** (2021). Covid-19 Pandemisi Boyunca Aşı Öncesi ve Aşı Sonrası Dönem için Karantinanın Enfekte Olmaya Etkisinin Oyun Teorisi Kullanılarak İncelenmesi, *Proceedings of Iğdır International Applied Sciences*, Iğdır Üniversity, pp.175–184.

[115] **İzgi, B.**, **Özkaya, M. and Üre, N.K.** (2022). Matrix norm yönteminin yapay zeka ile uygulanması, *Proceedings of the 2nd International Congress on Modern Sciences*, Tashkent Chemical-Technological Institute, pp.806–815.

[116] **Li, S.**, **Chen, M.**, **Wang, Y. and Wu, Q.** (2022). A Fast Algorithm To Solve Large-Scale Matrix Games Based On Dimensionality Reduction And Its Application In Multiple Unmanned Combat Air Vehicles Attack-Defense Decision-Making, *Inf. Sci.*, *594*, 305–321.

[117] **Puterman, M.L.** (2014). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, John Wiley & Sons.

[118] **Bellman, R.** (1957). A Markovian Decision Process, *Journal of Mathematics and Mechanics*, 679–684.

[119] **TowardsData**. *Reinforcement learning: Markov decision process Part 1*, https://towardsdatascience.com/policy-iteration-in-rl-an-illustration-6d58bdcb87a7, accessed: 2024-10-09.

[120] **Rawat, W. and Wang, Z.** (2017). Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review, *Neural Computation*, *29*(9), 2352–2449.

[121] **Wang, W.**, **Yang, Y.**, **Wang, X.**, **Wang, W. and Li, J.** (2019). Development of Convolutional Neural Network and Its Application in Image Classification: A Survey, *Optical Engineering*, *58*(4), 040901–040901.

[122] **Bochenek, B. and Ustrnul, Z.** (2022). Machine Learning in Weather Prediction and Climate Analyses: Applications and Perspectives, *Atmosphere*, *13*(2), 180.

[123] **Traore, B.B.**, **Kamsu-Foguem, B. and Tangara, F.** (2018). Deep Convolution Neural Network for Image Recognition, *Ecological Informatics*, *48*, 257–268.

[124] **Chen, S. and He, H.** (2018). Stock Prediction Using Convolutional Neural Network, *IOP Conference Series: Materials Science and Engineering*, *435*, 012026.

[125] **Özkaya, M. and İzgi, B.** (2024). Solution of Markov Reward Games Using Convolutional Neural Networks, *In the proceedings of International Conference on Applied Analysis and Mathematical Modeling 2024*, 210–211.

CURRICULUM VITAE

Name SURNAME: Murat ÖZKAYA

EDUCATION:

- **B.Sc.:** 2016, Abant İzzet Baysal University, Faculty of Arts and Science, Department of Mathematics
- **M.Sc.:** 2018, Istanbul Technical University, Faculty of Science and Letter, Department of Mathematical Engineering

PROFESSIONAL EXPERIENCE AND REWARDS:

- 2020 - Current. Research Assistant at Çanakkale Onsekiz Mart University.

PUBLICATIONS ON THESIS:

- **İzgi B., Özkaya M., Üre N. K., Perc M.** 2025. A Holistic Matrix Norm-Based Alternative Solution Method for Markov Reward Games, *Applied Mathematics and Computation,* 488, 12912.
- **İzgi B., Özkaya M., Üre N. K., Perc M.** 2024. Matrix Norm Based Hybrid Shapley and Iterative Methods for the Solution of Stochastic Matrix Games, *Applied Mathematics and Computation,* 473, 123638.
- **İzgi B., Özkaya M., Üre N. K., Perc M.** 2023. Machine Learning Driven Extended Matrix Norm Method for the Solution of Large-scale Zero-sum Matrix Games, *Journal of Computational Science,* 68, 101997.
- **İzgi B., Özkaya M., Üre N. K., Perc M.** 2023. Extended Matrix Norm Method: Applications to Bimatrix Games and Convergence Results, *Applied Mathematics and Computation,* 438, 127553.
- **Özkaya M. and, İzgi B.** 2021. Effects of the Quarantine on the Individuals' Risk of Covid-19 Infection: Game Theoretical Approach, *Alexandria Engineering Journal,* 60(4), 4157-4165.
- **İzgi B., Özkaya M. and Üre, N. K.** 2022. Matrix Norm Yönteminin Yapay Zeka ile Uygulanması, *2. International Congress on Modern Sciences*, ISBN: 978-625-8254-11-2, December 16-17, 2022, Tashkent, Uzbekistan.
- **Özkaya M. and, İzgi B.** 2021. Covid-19 Pandemisi Boyunca Aşı Öncesi ve Aşı Sonrası Dönem için Karantinanın Enfekte Olmaya Etkisinin Oyun Teorisi Kullanılarak İncelenmesi, *Iğdır International Applied Sciences*,ISBN 978-605-74616-9-8, April 14-15, 2021 Iğdır, Türkiye.

PRESENTATIONS ON THESIS:

- **İzgi B. and Özkaya M.** 2024. Hybrid Shapley Iteration for Stochastic Matrix Games, *International Conference on Modern Problems of Mathematics, Mechanics and their Applications*, June 20-22, Baku, Azerbaijan.

- **Özkaya M. and İzgi B.** 2024. Solution of Markov Reward Games Using Convolutional Neural Networks,*12. International Conference on Applied Analysis and Mathematical Modeling*, July 19-23, 2024 Istanbul, Turkey.

- **İzgi B., Özkaya M. and Üre, N. K.** 2022. Matrix Norm Yönteminin Yapay Zeka ile Uygulanması, *2. International Congress on Modern Sciences*, December 16-17, 2022, Tashkent, Uzbekistan.

- **Özkaya M. and, İzgi B.** 2021. Covid-19 Pandemisi Boyunca Aşı Öncesi ve Aşı Sonrası Dönem için Karantinanın Enfekte Olmaya Etkisinin Oyun Teorisi Kullanılarak İncelenmesi, *Iğdır International Applied Sciences*, April 14-15, 2021 Iğdır, Türkiye.