



CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS
AVANZADOS DEL INSTITUTO POLITÉCNICO NACIONAL

UNIDAD ZACATENCO
DEPARTAMENTO DE COMPUTACIÓN

**Selection of Strategies in Complex Games:
Baseball, American Football and Go**

A dissertation submitted by

Arturo Yee Rendón

For the degree of

Doctor in Computer Science

Advisor:

Dr. José Matías Alvarado Mentado

México, D.F.

Mayo, 2015



CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS
AVANZADOS DEL INSTITUTO POLITÉCNICO NACIONAL

UNIDAD ZACATENCO
DEPARTAMENTO DE COMPUTACIÓN

**Selección de estrategias en juegos complejos:
Béisbol, Fútbol Americano y Go**

Tesis que presenta

Arturo Yee Rendón

Para obtener el grado de

Doctor en Ciencias de la Computación

Director de tesis:

Dr. José Matías Alvarado Mentado

México, D.F.

Mayo, 2015

Dedico esta tesis a

Mi hija Maria Fernanda Yee Torres y

A mi esposa Maria Irene Torres Silvas

Abstract

Game theory (GT) is the science of strategic decision making, which is used to study the competition and cooperation relationships among entities. In multi-player games, GT is a powerful tool to study the decision-making, where players should make choices that affect the interests of other players, and the whole team. In this thesis, we designed and implemented mathematical formal models, based on Finite State Machine for the Baseball, American Football, and the game of Go. In addition, we implemented efficient methods for the strategic analysis, for Baseball and American Football, we use Nash equilibrium and Pareto efficiency, and for Go gaming automation, Neural Networks and Monte Carlo Tree Search.

The multi-player game modeling is of a high complexity, and the strategic analysis of Baseball and American Football must include a large number of parameters for fairly automatized decision-making support [5]. The strategic analysis for Baseball and American Football involves the use of Nash equilibrium and Pareto efficiency. These classical methods are used in economic, for analyzing the interactions among actors in a social dynamic context. In Baseball and American Football, they are used for modeling the strategic behavior of multiple players in a game match. To apply these methods, utility functions for the strategy profiles selection are constructed based on empirical data.

The computer simulations results of Baseball and American Football matches show that, although the usual *non-cooperative* qualification to Nash equilibrium, it is a relative adjective, up to the real circumstance. In the context of Baseball or American Football matches, where several parameters out of the players' and manager's control, Nash equilibrium allows to identify strategy profiles, for effective cooperation in real circumstances. The use of Nash equilibrium prevents to try plays or strategies with low statistical occurrence, thereby decreasing the risk to lose points, i.e., the Nash equilibrium strategy profiles, frequently include plays and strategies with high statistical occurrence, thus these strategies are more feasible in real matches. On the other hand, Pareto efficiency induces to choose the theoretically optimum strategy profiles, however we observe that the optimal plays and strategies have low statistical occurrence, therefore

they are impractical in real circumstances. The Pareto strategy profiles are less likely to occur than the Nash ones. The strategies in Pareto profiles may be the most profitable, but their probability of occurrences are low. The combined use of both, Nash and Pareto efficiency, for strategic choice on multi-player Baseball and American football games, it is relevant to circumstances with complex social interactions.

Nowadays, the formal analysis of the board game of Go is paradigmatic for computer science, since this game is a top complex game and currently, design and implement learning methods for Go gaming automation, conceal a huge combinatorial complexity and the challenge is to create efficient methods, that give good strategies for playing Go. Two major methods are analyzed and quantified for the Go gaming automation in this thesis, Neural Networks and Monte Carlo Tree Search. The Neural Networks are used for pattern recognition of Go tactics, and Monte Carlo Tree Search is used for evaluating the next move from a board configuration. The main remarks from this analysis is that, the use of Neural Networks for pattern recognition of Go *eyes*, *ladders* and *nets* is the best option in the early and middle stages of a Go match, and in the end stages, Monte Carlo Tree Search is the preferred method to be used.

Resumen

La teoría de juegos (GT por sus siglas en inglés) es la ciencia de la toma de decisiones estratégicas, la cual es utilizada para estudiar las relaciones de competencia y cooperación entre entidades. En juegos de multijugador, GT es una poderosa herramienta para estudiar la toma de decisiones, donde varios jugadores deben tomar decisiones que potencialmente afectan a los intereses de los otros jugadores. En esta tesis, diseñamos e implementamos modelos formales matemáticos, basados en máquinas de estado finito para el béisbol, fútbol americano y Go. Además, implementamos métodos eficientes para el análisis de estrategias, para el béisbol y el fútbol americano, utilizamos el equilibrio de Nash y la eficiencia de Pareto, y para el juego Go, las redes neuronales y Monte Carlo *Tree Search*.

El modelado de juegos de multijugador es de alta complejidad, y el análisis de estrategias de estos juegos, deben incluir un gran número de parámetros, para una correcta toma de decisiones [5]. Para el análisis de estrategias en el béisbol y en el fútbol americano, se usa el equilibrio de Nash y la eficiencia de Pareto. Estos métodos clásicos de la economía, son aplicados para el análisis de las interacciones entre actores en un contexto dinámico social. En el béisbol y en el fútbol americano, estos métodos son útiles para modelar el comportamiento estratégico, de varios jugadores en un partido. Para el uso de estos métodos, funciones de utilidad para la selección de perfiles de estrategias se construye utilizando datos empíricos.

Los resultados de las simulaciones por computadora de partidos de béisbol y de fútbol americano muestran que, aunque la calificación de no-cooperación habitual para el equilibrio de Nash, es un calificativo relativo, sobre las circunstancias reales. En el contexto de un partido de béisbol o de fútbol americano, con varios parámetros fuera del control de los jugadores y del manager, el equilibrio de Nash permite identificar perfiles de estrategias para una cooperación eficaz en circunstancias reales. El uso de equilibrio de Nash evita realizar jugadas o estrategias con baja incidencia estadística, lo cual disminuye el riesgo de perder puntos, esto significa que, los perfiles de estrategia de equilibrio de Nash con asiduidad incluyen jugadas y estrategias con incidencia estadística

grande, las cuales son más factibles de que ocurrirán en partidos reales. Por otro lado, la eficiencia de Pareto induce a elegir perfiles de estrategias teóricamente óptimos, pero observamos que las jugadas y estrategias óptimas tienen baja incidencia estadística, por lo que son poco probable de que ocurran en circunstancias reales de un partido de béisbol o fútbol americano. Los perfiles de estrategia eficiente de Pareto son menos probable que ocurra que los de Nash. Las estrategias en los perfiles de Pareto pueden ser los más rentables, pero sus estadísticas de ocurrencia son bajas. El uso combinado de ambos, Nash y Pareto eficiencia, para la toma de decisiones estratégicas en juegos de multijugador, es relevante en circunstancias de interacciones sociales complejas.

Hoy en día, el análisis formal del juego de tablero Go, es paradigmático en la ciencia de la computación, debido a que el Go es un juego muy complejo y en la actualidad, diseñar e implementar métodos de aprendizaje para la automatización del él, esconden una enorme complejidad combinatoria, por lo que el reto es la creación de métodos eficientes, que den buenas estrategias para jugar el juego de Go. En esta tesis, dos métodos principales son analizados y cuantificados, para la automatización del juego de Go, las redes neuronales y Monte Carlo *Tree Search*. Las redes neuronales son usadas para el reconocimiento de tácticas del juego de Go, tales como, *eyes*, *ladders* and *nets*, y Monte Carlo *Tree search* es usado para evaluar el siguiente movimiento desde una configuración del tablero. Las principales observaciones del análisis cuantitativo desarrollado es que, el uso de las redes neuronales durante las etapas iniciales e intermedias de un partido de Go es la mejor opción, y en etapas finales, Monte Carlo *Tree Search* es el método preferido a ser usado.

Agradecimientos

Deseo agradecer, muy especialmente a todo mi familia, a mis padres Manuel Yee y Manuela Rendón, les agradezco su amor y cariño, a mi hija María Fernanda y a mi esposa María Irene por su amor incondicional y apoyo para culminar esta etapa, a mi hermanos Cristo Manuel, Ana Julia, Zumey y Bruce por sus valiosos consejos, a mis sobrinos Sarah Sophia y Mateo, y a mis cuñados Raúl, Lorena y Jing.

Deseo agradecer al Departamento de Computación del Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional (CINVESTAV-IPN) y a la Universidad Autónoma de Sinaloa (UAS), por brindarme las facilidades y apoyos necesarios para el desarrollo del presente trabajo. Agradezco al CONACyT la beca otorgada durante la realización de estos estudios de doctorado.

Deseo agradecer a mí asesor, al Dr. Matías Alvarado, por aceptar dirigir el presente trabajo por su apoyo, sus conocimientos e ideas, y su dedicación para este trabajo.

Agradezco al Dr. Ulises Cortés, al Dr. Germinal Cocho, al Dr. Carlos Enrique Valencia, a la Dra. Sonia Mendoza, a la Dra. María Dolores Lara, y al Dr. Amilcar Meneses, por aceptar ser revisores de este trabajo, y por sus valiosos comentarios y observaciones.

Quiero agradecer a mis compañeros de doctorado, Alfredo Arias, Sandra Díaz, Lil María, Luis Miguel, Adriana Menchaca, Edgar Manoatl, Saúl Zapotecas, Cuauhtémoc Macillas, Adrián Sosa, Sergio Alvarado, Carlos Hernández y así como, a un buen amigo Azael y a todo los demás compañero del doctorado, con los cuales quienes conviví más de cerca durante estos últimos cuatro años. Gracias a todos ellos por los gratos momentos de convivencia, y por su disposición a compartir sus ideas en los ámbitos académico y personal.

También agradezco al personal del departamento de Computación, Sofía Reza, Felipa Rosas y Erika Ríos por su apoyo en diversos trámites académicos y administrativos. Gracias también a los auxiliares de investigación, José Luis Garcilazo, Santiago Domínguez y Arcadio Morales.

Y a todas aquellas personas, que de alguna manera estuvieron presentes en el desarrollo del presente trabajo, pero que por motivo de espacio sus nombres son omitidos.

Publications

The publications from this thesis are the following:

Papers

I. Published papers:

Journal papers

1. Matías Alvarado and Arturo Yee, "Nash equilibrium for collective strategic reasoning", **Expert System with Application**, 2012. **39**(15). Indexing in SCISEARCH, Science Citation Index, Scopus.
2. Matías Alvarado, Arturo Yee and Germinal Cocho, "Simulation of baseball gaming by cooperation and non-cooperation strategies", **Computación y Sistemas**, 2014. **18**(4). Indexing in CONACYT Index of Excellence of Mexican Journals, Scopus, Redalyc, E-Journal, e-revist@s, Latindex, Biblat, Periodica, DBLP, and SciELO.

Congress papers

1. Arturo Yee Rendón and Matías Alvarado, "Formal language and reasoning for playing Go", Proceedings of LANMR'11, published, 2011.
2. Arturo Yee and Matías Alvarado, "Pattern Recognition and Monte-Carlo Tree Search for Go Gaming Better Automation", Proceedings of IBERAMIA 2012, LNAI 7637, 2012.
3. Matías Alvarado, Arturo Yee and Jesús Fernández, "Simulation of American Football Gaming", Advances in Sport Science and Computer Science, (ISSN: 1743- 3517), 2014.
4. Arturo Yee, Reinaldo Rodriguez and Matías Alvarado, "Analysis of Strategies in American Football Using Nash Equilibrium", the proceedings of AIMSA 2014, LNCS 8722, 2014.

5. Arturo Yee and Matías Alvarado, "Methodology for the Modeling of Multi-Player Games", Proceedings of the 18th International Conference on Computers (part of CSCC '14), 2014.
6. Arturo Yee and Matías Alvarado, "Well-Time pattern recognition in Go gaming automation", Mathematical Methods and Computational Techniques in Science and Engineering, 2014.

II. Submitted Papers

1. Arturo Yee, Matías Alvarado and Germinal Cocho, "Simulation on best team formation and selection of strategies for baseball gaming", 2015, (under-review).
2. Arturo Yee and Matías Alvarado. "Simulation of strategic choices in an American football game", 2015, (under-review).
3. Matías Alvarado, Carlos Villareal, Sergio Camposortega and Arturo Yee, "Ising model for computer Go", 2015, (under-review).
4. Arturo Yee and Matías Alvrado, "Mathematical modeling and analysis of learning techniques for the game of Go", 2015, (under-review).

Patents

1. Simulador y módulo para la selección de estrategias al jugar Beisbol (expediente **MX/a/2015/002133**).

See, the summary of all my contributions of this thesis, in Section 5.2.

Contents

- 1 Introduction27
 - 1.1. Scope of the Thesis29
 - 1.1.1. The Problem29
 - 1.1.2. Hypothesis30
 - 1.1.3. Objectives31
 - 1.2. Baseball31
 - 1.3. American Football32
 - 1.4. The Game of Go.....33
 - 1.5. Content Organization.....34
- 2 Simulation of Baseball Gaming37
 - 2.1 Overview.....37
 - 2.1.1 The Prisoner’s Dilemma38
 - 2.1.2 Qualitative Analysis39
 - 2.1.3 The Strategic Nature of Baseball41
 - 2.2 Formal Language41
 - 2.3 The Finite State Machine.....44
 - 2.4 Generator of Plays46
 - 2.5 Selection of Strategies and Payoff Matrices.....49
 - 2.5.1 Nash Equilibrium for Non-Cooperation Strategies49
 - 2.5.2 Pareto Efficiency for Cooperation Strategies51
 - 2.5.3 Modeling of Payoff Matrices52
 - 2.5.3.1 Players’ Payoff Functions54
 - 2.5.3.2 Examples of Analysis of Strategies.....58
 - 2.6 Hungarian Algorithm for Baseball Team Selection.....61
 - 2.6.1 Data Normalization62
 - 2.6.2 Hungarian Algorithm.....65
 - 2.7 Merging Equilibrium of Nash and Pareto Efficiency: Test and Comparison.....66

| | | |
|-------|---|-----|
| 2.7.1 | Simulation using MLB Data or Selection of Strategies..... | 66 |
| 2.7.2 | The Mix Election Methods..... | 69 |
| 2.7.3 | Nash Offensive versus Pareto Efficiency Defensive | 78 |
| 2.8 | Hungarian Selection: Team Performance Comparison..... | 82 |
| 2.9 | Discussion..... | 86 |
| 2.9.1 | Nash Equilibrium Computing Complexity | 88 |
| 2.9.2 | Cooperation Management | 88 |
| 2.9.3 | Couple and Team Formation..... | 90 |
| 2.9.4 | Critical Zone Behavior in Biological and Physical Systems..... | 91 |
| 2.10 | Conclusion | 92 |
| 3 | Simulation of American Football Gaming..... | 95 |
| 3.1 | Overview | 95 |
| 3.1.1 | Description | 96 |
| 3.1.2 | Strategy Thinking: Cooperation and Non-Cooperation..... | 97 |
| 3.2 | Formal Language..... | 99 |
| 3.3 | Finite State Machine | 101 |
| 3.4 | Strategies by Team-Roles and the Average Occurrence of Plays | 103 |
| 3.4.1 | Offensive Team Plays..... | 105 |
| 3.4.2 | Defensive Team Plays..... | 105 |
| 3.4.3 | Special Team Plays | 106 |
| 3.5 | Setting-Up of Payoff Functions..... | 106 |
| 3.5.1 | Offensive Team | 107 |
| 3.5.2 | Defensive Team..... | 108 |
| 3.5.3 | Special Team | 108 |
| 3.5.4 | Examples | 109 |
| 3.6 | Selection of Strategies: Merging of and Experiments..... | 111 |
| 3.6.1 | Simulations using NFL Data or Selection of Strategies | 111 |
| 3.6.2 | Using Nash and Pareto Efficiency in Teams with Common NFL Statistics 114 | |
| 3.6.3 | Analysis of Results..... | 116 |

| | | |
|-------|--|-----|
| 3.6.4 | Score Forecasting..... | 119 |
| 3.7 | Discussion | 121 |
| 3.8 | Conclusion..... | 124 |
| 4 | The Game of GO | 127 |
| 4.1 | State of the Art..... | 129 |
| 4.1.1 | Simulation-Based Search..... | 130 |
| 4.1.2 | Neural Network for Learning/Recognition..... | 132 |
| 4.1.3 | Go Pattern Recognition..... | 134 |
| 4.1.4 | Evolutionary Algorithms | 135 |
| 4.1.5 | Planning Techniques..... | 137 |
| 4.2 | Mathematical Modeling | 137 |
| 4.3 | Algorithmic Modeling | 139 |
| 4.4 | Tactics Recognition and Strategies Building | 142 |
| 4.4.1 | Tactics Pattern Recognition | 143 |
| 4.4.2 | Building of Strategies from Pattern Recognition..... | 145 |
| 4.5 | Strategic Players | 147 |
| 4.5.1 | Hybrid SP_1 Player with MCTS..... | 148 |
| 4.5.2 | Hybrid SP_2 Player with <i>GNUGo</i> | 149 |
| 4.6 | Go Gaming Experiments and Comparisons | 149 |
| 4.7 | Players' Performance Analysis | 155 |
| 4.7.1 | SP_1 Performance Comparison..... | 155 |
| 4.7.2 | SP_2 Performance Comparison..... | 156 |
| 4.7.3 | NN-Based Approaches Comparison..... | 158 |
| 4.8 | Gaming Discussion | 161 |
| 4.9 | Conclusion..... | 163 |
| 5 | General Conclusion..... | 165 |
| 5.1 | Conclusions..... | 165 |
| 5.2 | Summary of Contributions | 167 |
| 5.2.1 | Methods..... | 167 |
| 5.2.2 | Products | 167 |

| | | |
|-------|---|-----|
| 5.2.3 | Impact in Media | 170 |
| 6 | Future work | 173 |
| 6.1 | Potential Analysis by Product of our Work | 173 |
| 6.2 | Ising Model for Computer Go..... | 173 |
| 6.2.1 | The Ising Model | 174 |
| 6.2.2 | Go Tactics by Common Fate Graph..... | 176 |
| 6.2.3 | Example of CFG | 177 |
| 6.2.4 | Go Modeling by Energy Function and CFG | 178 |
| 6.2.5 | Experiments and Results | 180 |
| | Appendix 2.A..... | 183 |
| | Bibliography..... | 185 |

Acronyms

| | |
|------------------|---------------------------------|
| AF | American Football |
| AS | Absolute Score |
| BPNN | Back Propagation Neural Network |
| CFG | Context-Free Grammar |
| DEN | Denver Broncos |
| FSM | Finite State Machine |
| GOBAN | Go Game Official Board |
| GT | Game Theory |
| HA | Hungarian Algorithm |
| KE | Kantian Equilibrium |
| MC | Monte-Carlo |
| MCTS | Monte-Carlo Tree Search |
| MLB | Major League Baseball |
| NE | Nash Equilibrium |
| NFL | National Football League |
| NN | Neural Network |
| NYY | New York Yankees |
| OAK | Oakland Athletics |
| OAK ¹ | Oakland Raiders |
| PE | Pareto Efficiency |
| RAVE | Rapid Action Value Estimation |
| RS | Relative Score |
| SO | Statistical Occurrence |
| SPs | Sacrifice Plays |
| TD | Temporal Difference |

List of Figures

| | |
|--|----|
| Figure 1.1 Go gaming flow diagram | 33 |
| Figure 2.1 Plays ordered. | 42 |
| Figure 2.2 Baseball FSM diagram. | 45 |
| Figure 2.3 Generation of baseball plays. | 47 |
| Figure 2.4 Probabilistic function scheme. | 48 |
| Figure 2.5 General scheme of the generation and construction of chains. | 48 |
| Figure 2.6 FSM for modeling the Nash or Pareto efficient strategies profiles of a team. | 52 |
| Figure 2.7 Matrix entries for a baseball quantitative analysis. | 53 |
| Figure 2.8 Two players' strategy profiles. | 53 |
| Figure 2.9 Payoff matrix for one player in a two-player game. | 54 |
| Figure 2.10 Entries for the payoff matrices of players 1 and 2 | 59 |
| Figure 2.11 Deviations in the strategy profiles. | 60 |
| Figure 2.12 T_1 NE vs T_2 NYY. | 68 |
| Figure 2.13 T_1 NE vs T_2 OAK. | 68 |
| Figure 2.14 T_1 PE vs T_2 NYY. | 68 |
| Figure 2.15 T_1 PE vs T_2 OAK. | 68 |
| Figure 2.16 Percentage of winning of T_1 vs. T_2 in item (1). | 72 |
| Figure 2.17 Percentage of winning of T_1 vs. T_2 in item (2). | 72 |
| Figure 2.18 Percentage of winning of T_1 vs. T_2 in item (3). | 72 |
| Figure 2.19 Percentage of winning of T_1 vs. T_2 in item (4). | 72 |
| Figure 2.20 Percentage of winning of T_1 vs. T_2 in item (5). | 72 |
| Figure 2.21 Percentage of winning of T_1 vs. T_2 in item (6). | 72 |
| Figure 2.22 Percentage of winning of T_1 vs. T_2 in item (7). | 73 |
| Figure 2.23 Percentage of winning of T_1 vs. T_2 in item (8). | 73 |
| Figure 2.24 Analysis of change of strategy technique of teams (3). | 74 |
| Figure 2.25 Analysis of change of strategy technique of teams (4). | 75 |
| Figure 2.26 Analysis of change of strategy technique of teams (5). | 76 |
| Figure 2.27 Analysis of change of strategy technique of teams (6). | 76 |

| | |
|---|-----|
| Figure 2.28 Analysis of change of strategy technique of teams (7). | 77 |
| Figure 2.29 Analysis of change of strategy technique of teams (8). | 78 |
| Figure 2.30 Percentage of winning of T_1 vs. T_2 in item (a). | 79 |
| Figure 2.31 Percentage of winning of T_1 vs. T_2 in item (b). | 79 |
| Figure 2.32 Percentage of winning of T_1 vs. T_2 in item (c). | 79 |
| Figure 2.33 Common strategic profiles between NE and PE. | 80 |
| Figure 2.34 Percentage of occurrence of NE profiles. | 80 |
| Figure 2.35 Percentage of occurrence of Pareto efficient profiles. | 81 |
| Figure 2.36 HA-NE <i>versus</i> NE. | 84 |
| Figure 2.37 HA-NE <i>versus</i> PE. | 84 |
| Figure 2.38 HA-PE <i>versus</i> NE. | 84 |
| Figure 2.39 HA-PE <i>versus</i> PE. | 84 |
| Figure 2.40 HA-NE <i>versus</i> HA-PE. | 85 |
| Figure 2.41 HA <i>versus</i> No-Method. | 85 |
| Figure 2.42 T_1 only uses (HA+NE) and T_2 uses (PE, NE, HA+PE, HA+NE). | 85 |
| Figure 2.43 T_1 only uses (HA+PE) and T_2 uses (PE, NE, HA+PE, HA+NE). | 86 |
| Figure 3.1 FSM for AF (a), (b), (c). | 103 |
| Figure 3.2 DEN stats <i>versus</i> OAK ¹ stats. | 113 |
| Figure 3.3 DEN stats <i>versus</i> OAK ¹ stats and NE. | 113 |
| Figure 3.4 DEN stats <i>versus</i> OAK ¹ stats and PE. | 113 |
| Figure 3.5 NE <i>versus</i> stats. | 115 |
| Figure 3.6 PE <i>versus</i> stats. | 115 |
| Figure 3.7 NE <i>versus</i> NE. | 115 |
| Figure 3.8 NE <i>versus</i> PE. | 115 |
| Figure 3.9 PE <i>versus</i> NE. | 116 |
| Figure 3.10 PE <i>versus</i> PE. | 116 |
| Figure 3.11 T_1 only uses stats and T_2 uses stats, PE and NE. | 117 |
| Figure 3.12 T_1 only uses stats and T_2 uses stats, PE and NE. | 118 |
| Figure 3.13 T_1 only uses PE and T_2 uses stats, PE and NE. | 118 |
| Figure 3.14 T_1 only uses NE and T_2 uses (stats, PE and NE). | 119 |

| | |
|---|-----|
| Figure 4.1 Go basic tactics: (a) eyes, unavailable points – A is black and B is white; (b) Ω stones are surrounded by a net and may soon be captured; (c) Ω stones are in atari by ladders. | 128 |
| Figure 4.2 Go basic strategies: (a) Ω stones perform invasion in territory dominated by white; (b) Ω black/white stones perform reduction in territory of white/black dominance; (c) black/white playing in positions A/B capture white/black stones; (d) black/white playing in positions A/B perform connections with friendly stones. | 128 |
| Figure 4.3 MCTS Approach, the process can be divided into: selection, expansion, simulation and back-propagation. | 130 |
| Figure 4.4 Multi-layer topology example. | 132 |
| Figure 4.5 FSM for modeling the player' strategies from basic actions α_x^i | 140 |
| Figure 4.6 The FSM modeling the α_x^i player's tactics. | 140 |
| Figure 4.7 FSM for eyes creation. | 141 |
| Figure 4.8 FSM for ladder creation. | 141 |
| Figure 4.9 FSM for net creation. | 141 |
| Figure 4.10 Go game process flow. | 142 |
| Figure 4.11 Illustrations of eyes, ladders and nets patterns. | 144 |
| Figure 4.12 Example pattern recognition of a possible eye. | 146 |
| Figure 4.13 Go gaming strategies: (a) Ω is in atari, but playing in point A makes a connection for saving Ω ; (b) playing in points A reduces Ω liberties; (c) white/black playing to points A/B increases dominance area; (d) white/black playing to points A/B interrupts the formation of adversary's eye. | 146 |
| Figure 4.14 Components of Go gaming simulator. | 147 |
| Figure 4.15 Hybrid Go gaming: pattern recognition of tactics and strategies by the early and middle match stages, and MCTS in the latter stages. | 148 |
| Figure 4.16 Tradeoff: number of neurons in the hidden layer versus the recognition percentage. | 150 |
| Figure 4.17 Number of patterns recognized through stages in a Go match. | 152 |
| Figure 4.18 Elapsed time per move throughout stages of a Go match. | 153 |

| | |
|--|-----|
| Figure 4.19 Percentage of random moves throughout stages of a Go match..... | 154 |
| Figure 4.20 First set of performance results of automated Go players. | 156 |
| Figure 4.21 Second set of performance results of automated Go players. | 157 |
| Figure 6.1 Phase transition by one stone placement on the Go board. | 175 |
| Figure 6.2 The representation of board configuration and CFG of Murakawa and Chao Chikun..... | 178 |
| Figure 6.3 Stones energy from Ising-model-based proposal to the match of Murakawa vs. Chao Chikun..... | 181 |
| Figure 6.4 Stones energy from Ising-model-based proposal to the match of Lee Changho vs. Ryu Suhang. | 182 |

List of Tables

| | |
|---|-----|
| Table 2.1 Prisoners' payoff matrix | 38 |
| Table 2.2 Σ = Terminals symbols to simple plays. | 42 |
| Table 2.3 Non-terminals symbols. | 43 |
| Table 2.4 Some grammar rules; H is used for hitting abbreviation. | 43 |
| Table 2.5 Transition function. | 46 |
| Table 2.6 Algorithm for baseball plays generation..... | 48 |
| Table 2.7 Nash equilibrium algorithm for selection of strategies in baseball gaming..... | 50 |
| Table 2.8 Pareto efficiency algorithm for selection of strategies in baseball gaming..... | 51 |
| Table 2.9 Summary of the analysis of strategy profiles | 60 |
| Table 2.10 Statistical measures to be considered. | 62 |
| Table 2.11 Some MLB baseball players' statistics. | 67 |
| Table 2.12 Examples of games scores in item (1)..... | 70 |
| Table 2.13 Examples of games scores in item (2)..... | 70 |
| Table 2.14 Examples of games scores in item (3)..... | 70 |
| Table 2.15 Examples of games scores in item (4)..... | 70 |
| Table 2.16 Examples of games scores in item (5)..... | 70 |
| Table 2.17 Examples of games scores in item (6)..... | 70 |
| Table 2.18 Examples of games scores in item (7)..... | 71 |
| Table 2.19 Examples of games scores in item (8)..... | 71 |
| Table 2.20 Player selected using Hungarian Algorithm..... | 82 |
| Table 2.21 The probability obtained for each player..... | 82 |
| Table 3.1 Σ = Terminals symbols. | 100 |
| Table 3.2 Non-terminals symbols. | 100 |
| Table 3.3 $R \subseteq (V - \Sigma) \times V^*$ some grammar rules..... | 100 |
| Table 3.4 Offensive and defensive plays..... | 104 |
| Table 3.5 SO of AF plays. | 104 |
| Table 3.6 Values of offensive plays by player. | 109 |
| Table 3.7 Values of defensive plays by player. | 109 |

| | |
|---|-----|
| Table 3.8 Forecasting game results using computer simulations | 120 |
| Table 4.1 Complexity of Go, Chess and Checkers games..... | 129 |
| Table 4.2 Results of NN eyes. | 150 |
| Table 4.3 Results of NN Ladders and nets. | 151 |
| Table 4.4 Elo Rating comparison..... | 158 |
| Table 4.5 Comparison of NN usage for playing Go. | 158 |
| Table 4.6 Analytical comparison among different NN approaches. | 159 |
| Table 4.7 Efficacy and Efficiency reported on the NN approaches. | 160 |
| Table 5.1 Information related to the impacts of newsletter..... | 170 |
| Table 6.1 Representation of the pattern of the stones for a CFG. | 176 |

1

Introduction

Game theory is the theory of independent and interdependent decision making. It is concerned with decision making in organizations, where the outcome depends on the decisions of two or more autonomous players, one of which may be nature itself, and where no single decision maker has full control over the outcomes. Games like Chess, Go, Baseball and American Football can be analyzed from a perspective of game theory [12, 28, 63, 72, 73, 82, 96].

There are, at least, three categories of games: games of *skill*, games of *chance*, and games of *strategy*. Games of skill are one-player games, whose defining property is the existence of a single player, who has complete control over all the outcomes [63]. Games of chance are those in which, the outcomes are at least partly determined by random factors rather than strictly by strategies. Unlike games of skill, the player does not control the outcomes completely, and strategic selections do not lead inexorably to certain outcomes. The outcomes of a game of chance depend partly on the player's choices, and partly on nature. Games of chance are further categorized as either involving risk or involving uncertainty. In the former, the player knows the probability of each of nature's responses, therefore knows the probability of success for each of his or her strategies. In games of chance involving uncertainty, probabilities cannot meaningfully be assigned to any of nature's responses [28], thus the player's outcomes are uncertain, and the probability of success unknown, some examples are: Poker, Craps, Roulette, among others.

Games of strategy are games involving two or more players, each of whom has partial control over the outcomes. In a way, since the players cannot assign probabilities to each other's choices, games of strategy are games involving uncertainty. They can be sub-divided into two-player games and multi-player games. Within each of these two sub-divisions, there are three further sub-categories depending on the way in which the pay-off functions are related to one another – whether the player's interests are completely coincident; completely conflicting; or partly coincident and partly conflicting [63], some examples are: the game of Go, Baseball, American Football, among others.

Games of strategy, whether two-player or multi-player, in which the players' interests coincide, are called *cooperative games of strategy*.

Games in which the players' interests are conflicting (i.e. strictly competitive games) are known as *zero-sum games of strategy*, so called because the pay-offs always add up to zero for each outcome of a fair game, or to another constant if the game is biased.

Games in which the interests of players are neither fully conflicting nor fully coincident are called *mixed-motive games of strategy*.

In this thesis, one problem to be considered is the formal modeling of complex multi-player games. There are a wide variety of multi-player games examples; therefore we try to cover many of these games in this thesis, by following the properties of multi-player games. The properties of multi-player games follow.

- 1. Alternating move games** are a kind of games in which, every time only one player can move, and the turn changes from one player to another in a predefined order in the game.
- 2. Simultaneous move games** are a kind of games in which all players have to move in each turn.
- 3. Impartial and partial game** are alternating move games in which, the legal moves depend only on the position of the game, and not on which player is currently moving, and the payoffs are symmetric. All games which are not impartial are partial.

Find a single game, which holds all multi-player properties, it is not possible, thus we chose strategically, Baseball and American Football, which are simultaneous move games, and Go, which is an alternating move game and impartial game.

1.1. Scope of the Thesis

This section describes the problem to solve on this doctoral work, the hypothesis and main objectives to address. Brief descriptions of the importance of the formal modeling on Baseball, American Football and Go are also included.

1.1.1. The Problem

In the broad context of modeling for system design, it is normally to assume that all decision-makers cooperate fully and therefore avoid conflict, however this is not always possible; in which case, the design process is best modeled and studied as a multi-player game.

The multi-player game modeling is of a high complexity, because a large number of game aspects, that have to be considered for doing a suitable model. The strategic analysis of these kinds of games must include a large number of parameters for fairly automatized decision-making support.

In Baseball and American Football, the formal modeling carries out a huge complexity; in a match, there are many uncertain factors like, human ways of pitching, batting, catching, passing and running, or natural factors like, wind speed, or the height of the place, and many others, which affect the game, and have to be considered for an appropriate modeling. For strategic analysis, many parameters must be considered like, the player playing style, or the circumstances of the match; in order to decide a suitable strategy, since it is crucial to obtain the match success.

In Go game, determining the next moves to perform, during an automated Go match is a hard problem, because the huge search space to assess. The complexity of computer Go gaming is measured by, the game tree size, and the state space, and it is quite de-

scriptive. The size of the Go game tree is around 10^{360} , 10^{172} for the state space, and up to 361 legal moves [6], therefore the search space on Go gaming solutions is huger (very much) than the one for Chess [35].

Based on the previously indicated difficulties, the major problems are: design and implement formal models for multi-player games, as well as, strategic analysis methods. The games to be considered are: Baseball, American Football and Go. Some relevant questions are the following:

1. How to design and implement computational models, for the strategic analysis on Baseball, American Football, and Go?
2. How to design and implement formal mathematical models, for Baseball, American Football, and Go?
3. Which are the game parameters from the match, or from the players, or others, that we need to consider, for a strategic analysis on multi-player games?
4. How to measure the teams' performance and the automated player's performance?

1.1.2. Hypothesis

Suitable computational models, using formal grammars and finite state machines, will be useful for a strategic analysis on multi-player games, since they will ensure the correct modeling of the game, covering many aspects for real-life and supported by real statistics. An appropriate analysis of the circumstances of a match will help to make the right decisions.

- Finite state machines and formal grammars, will help to design the correct computational models for Baseball, American Football, and the game of Go.
- The use of the traditional method in game theory, like Nash equilibrium and Pareto efficiency, for selection of strategies either alone or in combination will improve the team performance in victories for Baseball and American Football.

- The use of pattern recognition on stages, where board is not so occupied, and Monte Carlo Tree Search for closing a match, will improve the performance of Go automated player.

1.1.3. Objectives

The present work aims to develop formal models and strategic analysis methods for multi-player games, which include the algorithmic and mathematical models for Baseball, American Football, and Go.

- To design and implement computational game models, which are useful for strategic analysis of Baseball, American Football and Go.
- To design and implement mathematical models for multi-player games, Baseball, American Football and Go.
- To advance the state of the art for multi-player games analysis, by developing efficient methods to analyze the strategies for multi-player games, which allow modeling the person's or group's behavior.
- To validate the performance of the proposed methods for strategic analysis, by verifying the success of team and automated players on test sets victories.

1.2. Baseball

Baseball is a top strategic team game, playing by two teams, bat-and-ball play in a diamond field. Each team is 9 players and usually a match lasts 9 innings, nevertheless, if there is not a winning team at the ninth inning, additional innings are allowed until one team wins. The team that gets more runs throughout the innings is the winner; runs are scored by offensive team players, by batting the ball and moving from the home plate to first, second and third base and back to home plate without being out by the defense team.

For team success, in a match, the design and use of team strategies as a positive combination of individual strategies are requisites. Nowadays, the decision making is being

supported by computer tools, for strategic analysis on single or multi-player games. GT was formally founded by Von Neumann and Morgenstern in 1944 [82], having his major purpose in the economics modeling for markets, trade and financial issues; currently, it has been applied to a growing number of areas such as, politics, biology, sociology, as well as, to information technologies and engineering.

1.3. American Football

American Football is one of the top strategic games, played by two teams on a rectangular shaped field, 100 yards long by 53.3 yards wide, with goalposts in the end of the field. Each team has 11 players and a match lasts 1 hour divided in four quarters. The offensive team goal is advance an oval ball, by running or passing toward the adversary's end field [7, 22, 50]. The ways to obtain points are by advancing the ball, ten yards at least, until reach to the end zone for touchdown scoring, or kicking the ball such that it passes in the middle of the adversary's goalposts for a field goal, or by the defensive tackling the ball carrier in the offensive end zone for a safety. The offensive team should advance the ball at least ten yards in at most four downs (opportunities) to obtain four additional downs; otherwise the defensive team that is avoiding the ten yards advance, changes to the offensive role. The current offensive team advance starts from the last ball stop position. If the defensive catches the ball before a down is completed, it starts the offensive role at this position. A down ends by the most common circumstances that follow: when a pass is not successful, or when a player is tackled inside the field, or when a ball gets off the field.

Recently, the formal modeling and strategic analysis for support the matches gaming of multi-player sports such as, American Football (AF) or Baseball are growing [8, 52, 104]. These multi-player games have led investigations in areas of sport science [2, 61, 70], computer science, game theory [3], operation research [8, 104], simulation models [37, 52], among others.

1.4. The Game of Go

The formal analysis of the board game called “Go” is at the core of advances in computer science, in the same way the analysis of Chess was during the 20th century [71]. Go is a top complex board game and currently, design and implement learning methods for Go gaming automation are central challenges for computational intelligence, to demonstrate sufficient skill to beat the top human Go masters. The Go game official board (GOBAN) is a 19 × 19 grid for two players using black-stones versus white-stones with zero-sum, deterministic, and perfect information [35]. By turn, each player places one black/white stone on one empty intersection or point of the board. Black plays first and white receives a compensation *komi*, by playing the second turn [13]. The goal of Go gaming is to control as much of the board area as possible, by means of complex strategies, applied through simple Go rules. Figure 1.1 presents the flow diagram for Go gaming.

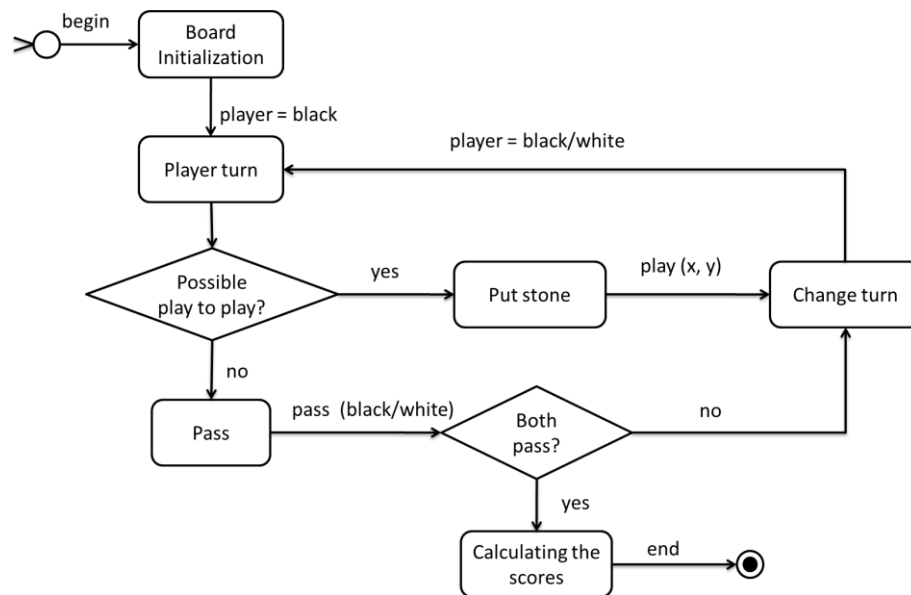


Figure 1.1 Go gaming flow diagram

To determine the available moves during an automated Go match is a tough problem because of the huge search space to assess. The complexity of computer Go gaming is

measured by the game tree size and the state space is quite descriptive. The game tree is the cardinality of the set of all possible manners for a legal sequence of moves, through all Go matches. A Go gaming state (node) is a particular board arrangement, i.e., the positions of the stones on the board at a specific moment in a match. The size of the Go game tree is around 10^{360} , 10^{172} for the state space and up to 361 legal moves [6]. Therefore, the search space on Go gaming solutions is huger –very much– than that for Chess [35]. The moves of a Go match are depicted graphically by a *decision tree* that records the moves and is an element of the game tree. The root state is the match beginning. Any node children are those positions reachable in one move.

On the Go board, the *liberty* of a stone is any vertical or horizontal unfilled point adjacent to the stone, which sometimes can be shared with other stones. Once a stone is placed on the board it can be removed only when it is captured, which happens if it is surrounded by adversarial stones and thus, losing all its liberties. Black stones capture white stones and vice versa. Two or more stones of the same color joined by horizontal or vertical points form a *chain stone* that cannot be divided; diagonally adjacent stones are not in a chain. From now on, the term stone refers to both single stones and chains, however, explicit differentiation is made when required. Any *stone is alive* if it cannot be captured, and it is *dead* if it cannot avoid capture. When a player places a stone that will result in immediate capture, this is called suicide, which is not allowed. The game ends when both players pass on their turn. Then, the score is computed based on both territory occupied by the player on the board and the number of captured adversarial stones. The winner is whoever has the largest total.

1.5. Content Organization

In Chapter 2, we describe all the analysis for Baseball game, the game review, the algorithms proposed, the set of tests, discussion and conclusion. In chapter 3, we present the formal analysis for American Football game, the game review, the formal grammar and finite state machine, discussion and conclusion. In chapter 4, we describe the Go game analysis, the state of art, the mathematical and algorithmic modeling, the experi-

ments, discussion and conclusion. In chapter 5, we present the general conclusion and the summary of the entire thesis. Finally, in Chapter 6, we present the future work.

2

Simulation of Baseball Gaming

2.1 Overview

Baseball is a zero-sum multi-player game that victory is entirely based on the appropriate strategies being practiced. Actually, this world popular team game is strategies thoughts obligated for playing [15], and the strategic decision-making is crucial to obtain the match success [72]. The strategies, as a set of organized plays are indicated by the team's manager regarding the specific profile of all the players hence each one's potential actions, as well the specific match circumstance; from all mentioned information the manager could select some strategies amid to obtain the most benefit. Baseball is at the time, cooperative from manager's perspective and non-cooperative from players' perspective: team's members are encouraged to act individually, but must cooperate for team's benefit too.

Nash Equilibrium (NE) concept allows typifying a team's strategy such that no player individually deviates by the selected collective strategy, because it will be prejudice for the player [81]. The concept of Pareto efficiency (PE) is widely used in the normative economics literature. The Pareto principle and unanimity are foundational in discussions of social welfare, social welfare functions, and social choice [88]. We say that an outcome is Pareto optimal if there is no other outcome in which all players are better off and at least one player is strictly better off. In our approach, the use of NE or PE is for identifying the team's strategies to apply throughout a Baseball match, either on offense

or defense role, for pushing up the probabilities of success by NE/PE-strategies-based playing, each Baseball player's potential actions is according to own profile given the current position at field, joint to the other players' profile and positions, in addition to the innings, number of outs and score, all integrated for supporting the manager's decision making at any moment during the match. Baseball NE/PE formal modeling is by means of payoff matrix. The computer Baseball simulator is based on: one formal grammar (set of rules), one FSM to recognize the language generated by the grammar, one random generator of Baseball plays, Hungarian algorithm for team selection, and for the analysis of strategies NE/PE algorithm for finding the best match collective strategies.

2.1.1 The Prisoner's Dilemma

There are circumstances where the players can get a better result cooperating, as illustrates in the prisoner's dilemma that describes when the police capture P_1 and P_2 , as suspicious people of a crime, without sufficient evidence to charge any of them. Questioned by the police, the possible both prisoners' dilemma strategies profiles to answer can be: (*silence, silence*), (*silence, confess*), (*confess, silence*) and (*confess, confess*). Separately, police offer the same deal: if one confesses but the accomplice not, the accomplice is ten years jail sentenced but confessor is released, so (3, 0) or (0, 3); if both silence (deny it) all the police can do is locking them up for six months due to a misdemeanor charge, (1, 1); if both confess five years jail is for each one (2, 2). The summary of prisoner's dilemma and the payoff matrix are shown in Table 2.1.

Table 2.1 Prisoners' payoff matrix

| Strategies | | P_2 | |
|------------|----------------|---|---|
| | | <i>Silence</i> | <i>Confess</i> |
| P_1 | <i>Silence</i> | $P_1, P_2, 6$ month jailed Profile (1, 1) | P_2 release, P_1 10 year jailed. Profile (0, 3) |
| | <i>Confess</i> | P_1 release, P_2 10 year jailed. Profile (3, 0) | $P_1, P_2, 6$ year jailed. Profile (2, 2) |

Condition to observe for a NE profile is that if a prisoner drawn up own strategy is for losing, because any NE strategy profile should maximize both player's profits. So, (*confess, confess*) is NE profile for both prisoners as long as any rational prisoner not deviate from, except to the risk to be negative affected: by keeping silence a prisoner can be twice jailed that if confesses.

In Prisoner's Dilemma games, a compensation mechanism where all players' payments pairs with mutual cooperation, it fits NE [32]; mutual cooperation is substantially more likely with payment pairs that bring the payoffs closer together, but if these payments are not permitted cooperation is much less likely. The players' mutual advantages by cooperation in Baseball are next analyzed.

2.1.2 Qualitative Analysis

In a strategy game, the intelligent planning mostly allows driving to victory. Baseball is a multi-player top strategic game, bat-and-ball play at a field [15, 17, 106]. Team is compound by 9 players and the match is 9 innings (high/low) initially, but if there is not winner at the ninth, additional innings are allowed. The game basic rules by the offensive are simple [38]: the offensive team's members take turns at the bat for attempting to hit the ball so thrown pitched from some distance and locating it away from adversaries in front of home plate. The runs are scored by the offensive team when a player, after bat ball sequentially advances from home plate to first, second and third base then back to home plate without being out by the defense team. The team scoring most runs throughout all the innings gets the victory. The team continues at the bat until three outs are made by the defensive team then switch the offense / defense team's roles.

The main offensive strategy is the appointment of the batting order before the game start, so the team's manager does the 9 players pre-set positions at bat. Usually, the best players are first at bat for having more opportunities to hit than those at the list end. Furthermore, at first two places put quick legs people trying as simple as possible to get them into the bases; then, the best hitters on 3rd and 4th position trailers to home with a home run or a good hit to give the players on bases enough time to move forward. In

addition, if one or more runners stay on any base a relevant offensive strategy is to intent to advance the runners, either by base stealing or by connecting a hit. If there are fewer than two outs, the sacrifice-plays-based strategy to advance runners is an option though could involve an out.

While the team at bat is for trying score runs, defensive team is for attempting record outs; the best defense strategy is to get the more outs as possible hence do not receive too many pitches and limits the opposing team's moves. A qualitative analysis of Baseball strategies, particularly the conservative – aggressive tradeoff, by regarding the score versus the inning order and number of outs result in the next alternatives: [106].

Score: Up on the scoreboard, play more recklessly can be a good option in order to increase the difference on the scoreboard; but down on the scoreboard, a conservative play is recommended, to keep the number of outs and the players on base.

Innings: Aggressive or conservative playing depends if the game is on initial, middle, or late innings. The first innings main goal is going-ahead by means of an aggressive playing without wasting outs by sacrifice bunts. Middle innings often determines the game character, in front of an aggressive style to play conservative is recommended. In the late innings and up on the scoreboard an aggressive play is recommended, but down on the scoreboard, a conservative play is better option.

Numbers of outs: Without outs, a small difference on the scoreboard and in the late innings, a conservative play should be applied for scoring few more runs and having one or two outs play aggressive to reach at least one run more.

Sacrifice Plays (SPs) are suitable strategies to lead the team victory, according to Baseball statistical study, by having less than two outs and a player at third base [68]. In well identified Baseball circumstances SPs are the best options, particularly the stolen base for a base-runner advances. Actually, sometimes, SPs correspond to NE strategy profiles [4].

2.1.3 The Strategic Nature of Baseball

In a competitive game the strategies planning looks for driving to victory. Baseball is a multi-player strategic game, bat-and-ball play at a field [15, 17, 106]. Each team is compound by 9 players and the match is 9 innings as usual, but if there is not winner at the ninth, additional innings are allowed. The offensive team's members take turns at the bat for attempting to hit the ball so thrown pitched from some distance and locating it away from adversaries in front of home plate [38]. The runs are scored by the offensive team when a player, after bat ball sequentially advances from home plate to first, second and third base then back to home plate without being out by the defense team. The team scoring most runs throughout all the innings gets the victory. The team continues at the bat until three outs are made by the defensive team then switches the offense/defense team's roles. The main offensive strategy is the appointment of the batting order before the game start: The best batters are first for having more opportunities to hit and take in the bases; furthermore, at first two places put quick legs people then the best hitters on 3rd and 4th position trailers to home with a home run or a good hit so the players on bases have enough time to move forward; in addition, to intent to advance the runners, either by base stealing or by connecting a hit, or apply a sacrifice-plays-based strategy to advance runners even it involves an out [68].

2.2 Formal Language

Whether for defense or offense role, the abbreviation to design simple and compound Baseball plays made by i player are listed in Table 2.2. Plays are weighted and total ordered based on their statistical occurrence (SO) in real life matches as Figure 2.1 shows: e.g. strikes occur more often than home runs, and balls occur more often than double plays.

| |
|--|
| $s^i \geq b^i \geq f^i \geq co^i \geq o^i \geq p^i \geq ce \geq hi^i \geq a1^i \geq wb^i \geq a2^i \geq d^i \geq dp^i \geq a3^i \geq a4^i \geq ca^i \geq r^i \geq fs^i \geq h^i \geq tb^i \geq bp^i \geq bg^i \geq w^i \geq tp^i \geq t^i \geq bo^i$ |
|--|

Figure 2.1 Plays ordered.

Notice that *wb* does not correspond to a typified Baseball play but is needed for a right modeling movements of players on bases, whose should wait the batter's actions hence properly play, e.g. if the batter's hit is a home-run (*h*) then the runners should go through bases to home, or if the batter gets out (*o*) the runners should stay on bases. For an expressive algorithmic implementation the Baseball rules are CFG (context-free grammar) design to facilitate the control of the number of strikes, balls, fouls or outs then the transition for modeling plays that occur after a given number of any of them. But it is a remark that a regular grammar, less expressive hence more complex than a CFG, is enough for modeling a whole Baseball match. CFG non-terminal elements that correspond to the names of single plays are listed in Table 2.3. The grammar rules generate the formal language (set of strings over the alphabet of symbols of simple plays) describing the whole Baseball match, see some of them in Table 2.4; rules ensure the correct composition of sentences describing the Baseball plays and matches. The Baseball CGF is as follows:

- V is the alphabet of terminals and non-terminals.
- $\Sigma \subseteq V$ is the set of terminals.
- $V - \Sigma$ is the set of non-terminal elements.
- $R \subseteq (V - \Sigma) \times V^*$ is the set of rules.
- $B \in V - \Sigma$ is the initial symbol.

Table 2.2 $\Sigma =$ Terminals symbols to simple plays.

| | | |
|------------------------|--------------------------|-------------------------------|
| b^i : ball | co^i : contact of ball | wb^i : wait batter's action |
| bo^i : bolck | h^i : homerun | a_1^i : move to A_1 |
| bg^i : base hit | hi^i : hit | a_2^i : move to A_2 |
| bp^i : base on balls | r^i : stealing base | a_3^i : move to A_3 |

| | | |
|---------------------------------|-------------------------------|--|
| d ⁱ : doublet | s ⁱ : strike | a ₄ ⁱ : move to home |
| f ⁱ : foul | t ⁱ : triple | ce: team change |
| dp ⁱ : double play | tb ⁱ : bunt | o ⁱ : out |
| fs ⁱ : sacrifice fly | tp ⁱ : triple play | p ⁱ : punched |
| | w ⁱ : wild pitch | |

Table 2.3 Non-terminals symbols.

| | |
|-----|---------------------------------------|
| A: | Action by ball contact |
| B: | Bat |
| B3: | Bat with three <i>outs</i> |
| M: | Movement |
| MH: | Home run movement |
| MR: | Stolen base movement |
| MG: | Base hit or base on movement by balls |
| MD | Movement by doublet |
| MT | Movement by triplet |
| R: | Steal |
| T: | Transition |

Table 2.4 Some grammar rules; H is used for hitting abbreviation.

| | |
|--|---|
| B -> b ⁱ B | H lead to ball, and hit back |
| B -> bp ⁱ MG B | H generate base on balls, making M and H return (4 balls later) |
| B -> s ⁱ B | H generate a strike and hit back |
| B -> p ⁱ B | H lead punch and hit back (3 strike later) |
| B -> p ⁱ B3 | H lead punch and hit back with three out (3 strikes and 2 outs later) |
| B -> f ⁱ B | H generate a foul, hitting back |
| B -> d ⁱ MD B | H generate a double, moving back to bat |
| B -> t ⁱ MT B | H generate a triplet, M and hit back |
| A -> hi ⁱ M B | Action to generate a hit, moving and re - bat |
| A-> o ⁱ B | Action to generate one out, H back |
| B -> h ⁱ MH B | H a home run generate M and hit back |
| B -> tb ⁱ M B | H generate a bunt, moving and return to bat |
| B -> tb ⁱ M o ⁱ B | H generate a bunt, moving, out and return to bat |
| B -> tb ⁱ M o ⁱ B3 | H generate a bunt, moving, out to bat and team change (2 outs later) |
| B -> w ⁱ M B | H generate a wild pitch, moving and hit back |
| B -> bg ⁱ MG B | H generate a base hit, moving and return to bat |
| B-> bo ⁱ M B | H generate a <i>bolk</i> , M and return to bat |
| B -> fs ⁱ M o ⁱ B | H generate a sacrifice fly, M, and back out to bat |

| | |
|--|--|
| B -> fs ¹ M o ¹ B3 | H generate a sacrifice fly, M, and change out of equipment |
| B3-> ce B | Hit three out, is change of equipment |
| ... | |
| $i \neq j, i \neq k, i \neq l, j \neq k, j \neq l, k \neq l$ | |

2.3 The Finite State Machine

A finite state machine (FSM) is a mathematical device for reading the input strings from a formal language; the parsing of strings starts at the FSM initial state then following through intermediate states; whenever the output or parsing end of a string occurs in a FSM halt state conclusion is that the string belongs to the language being recognized by this FSM [103]. A so called Alan Turing FSM deserves for reading a recursive enumerable –the most general formal– language by means of the corresponding transition function; the FSM memory containing the language is represented by a strip of tape [11]. A push-down machine, less general than the Allan Turing one, allows recognizing the CFG language describing both the simple or complex Baseball plays [11, 57].

Let $(\Sigma, \hat{S}, s_0, \varphi, H)$ be a push-down FSM such that:

- Σ is the alphabet.
- $\hat{S} = \{s, s_0, s_1, s_2, s_3\}$ is the set of states.
- $\varphi: \hat{S} \times \Sigma \rightarrow \hat{S}$ is the transitions function.
- $s_0 \in \hat{S}$ is the initial state.
- $H = \{s, s_0\} \subseteq \hat{S}$ is the set of halt states.

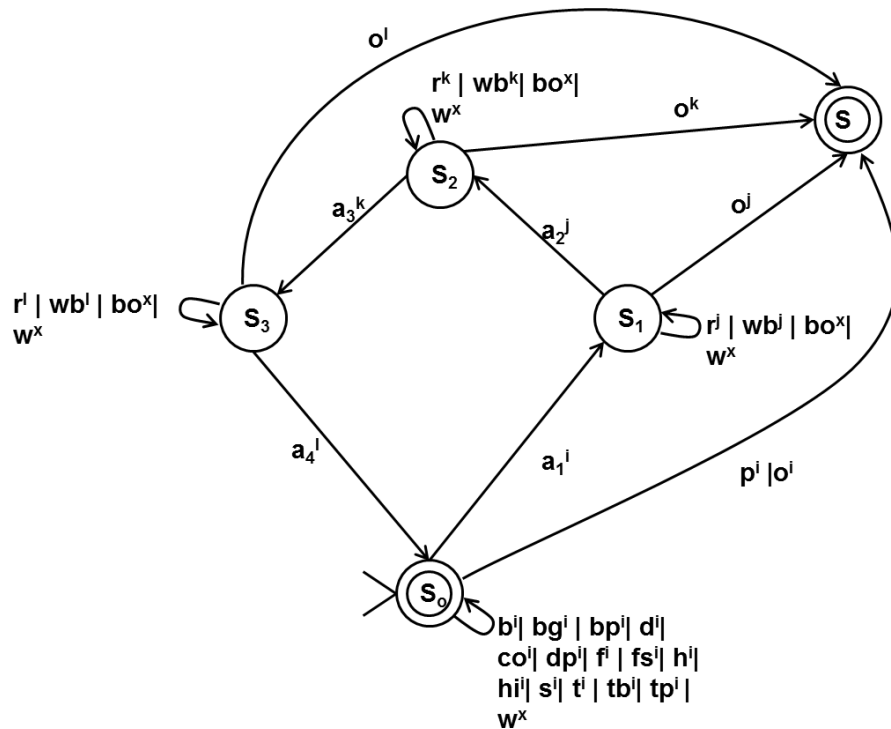


Figure 2.2 Baseball FSM diagram.

The FSM for Baseball is shape-of-field-like modeled: the home, 1st, 2nd and 3rd bases and a special base, these bases are modeled as the FSM states; the transitions between states (one to one) are the plays (movements) the players can perform, see Baseball FSM in Figure 2.2 and in Appendix 2.A, e.g. hits, fouls, strikes, among others. Stacking and de-stacking symbols to/from the FSM stacks are for respective scoring the number of strikes ST , fouls F , balls B , outs O , and the players on the bases, see transition function in Table 2.5. Strings that end in s_0 are run-strings and those that end in s are out-strings. By starting with the empty string (ε) each next play is right concatenated, the super-indexed is for indicating the player who performs the play. Plays being dependent on prior ones are generated whenever the need sequence of previous plays have occurred, e.g., for a base on balls must happen four previous balls, or for a punch must have three previous strikes.

Table 2.5 Transition function.

| | |
|---------------------------------------|--------------------------------|
| $(s_0, f, nil) : (s_0, F')$ | $(s_1, r, nil) : (s_1, nil)$ |
| $(s_0, s, nil) : (s_0, ST')$ | $(s_1, wb, nil) : (s_1, nil)$ |
| $(s_0, b, nil) : (s_0, B')$ | $(s_1, bo, nil) : (s_1, nil)$ |
| $(s_0, bp, nil) : (s_0, nil)$ | $(s_1, w, nil) : (s_1, nil)$ |
| $(s_0, bg, nil) : (s_0, nil)$ | $(s_1, o, nil) : (s, O')$ |
| $(s_0, bo, nil) : (s_0, nil)$ | $(s_1, a_2, a_1) : (s_2, a_2)$ |
| $(s_0, d, nil) : (s_0, nil)$ | $(s_2, r, nil) : (s_2, nil)$ |
| $(s_0, hi, nil) : (s_0, nil)$ | $(s_2, wb, nil) : (s_2, nil)$ |
| $(s_0, h, nil) : (s_0, nil)$ | $(s_2, bo, nil) : (s_2, nil)$ |
| $(s_0, fs, nil) : (s_0, nil)$ | $(s_2, w, nil) : (s_2, nil)$ |
| $(s_0, t, nil) : (s_0, nil)$ | $(s_2, o, nil) : (s, O')$ |
| $(s_0, tb, nil) : (s_0, nil)$ | $(s_2, a_3, a_2) : (s_3, a_3)$ |
| $(s_0, w, nil) : (s_0, nil)$ | $(s_3, r, nil) : (s_3, nil)$ |
| $(s_0, a_1, \{F ST B\}) : (s_1, a_1)$ | $(s_3, wb, nil) : (s_3, nil)$ |
| $(s_0, p, \{F ST B\}) : (s, O')$ | $(s_3, bo, nil) : (s_3, nil)$ |
| $(s_0, o, \{F ST B\}) : (s, O')$ | $(s_3, w, nil) : (s_3, nil)$ |
| $(s_0, dp, \{F ST B\}) : (s, O')$ | $(s_3, o, nil) : (s, O')$ |
| $(s_0, tp, \{F ST B\}) : (s, O')$ | $(s_3, a_4, a_3) : (s_4, a_4)$ |

2.4 Generator of Plays

The generator of Baseball plays produces strings that must have a correct sequence of moves, i.e., the Baseball plays should generate according to their average SO in real life games and the sequence should be consistent with reality. A generator of plays is useful, because it generates valid Baseball strings randomly, quickly and easily.

The generator produces Baseball plays and verifies that:

- These must be made based on their average SO, and also
- Be generated following the rules of the game.

Figure 2.3 shows generation of Baseball plays, so from the $\{1, \dots, m\}$ simple plays the complex ones are compound. In order the strings describing a simple or complex Baseball play preserve the condition of random occurrence—as in real life matches—, a random numbers generator is associated to determine any play occurrence. Posterior, the Baseball FSM validates the correctness of the arrangements of plays.

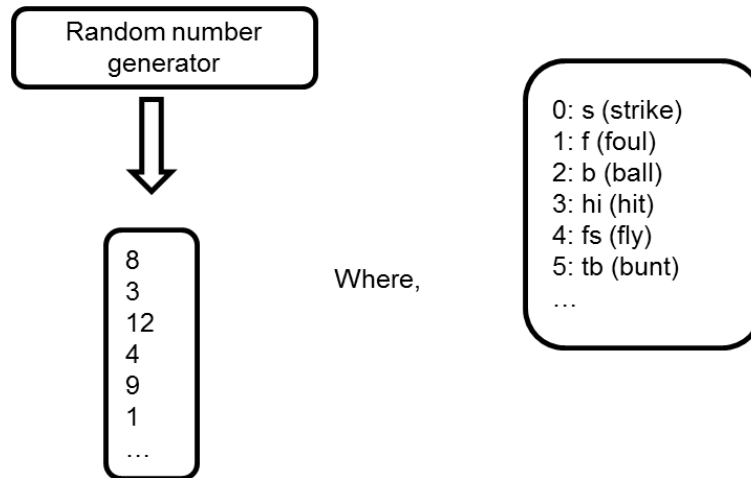


Figure 2.3 Generation of baseball plays.

In addition, the condition of real SO of a validated play is modeled by means of a flip function, it receives the SO of the play and return a random output [0, 1], which means to play or not the play, so, greater is the SO more 1s occurrence and conversely; if the SO is 0.5, it will return 1 or 0 with the same frequency, see how the flip function is used in Figure 2.4.

The generator has a module for generation and validating of strings. Once having the Baseball play to perform, this has to concatenate with the previous plays. The way to do this is as follows: at the right end of a string, empty one (ϵ) in the beginning, the play is concatenated with the previous ones and also indicating the player who performs it. The diagram for creating Baseball complex but right SO of plays is depicted in Figure 2.5; the algorithm for Baseball plays, since basic ones to a whole match according to real SO is in Table 2.6. Computer simulator implementation is in C language.

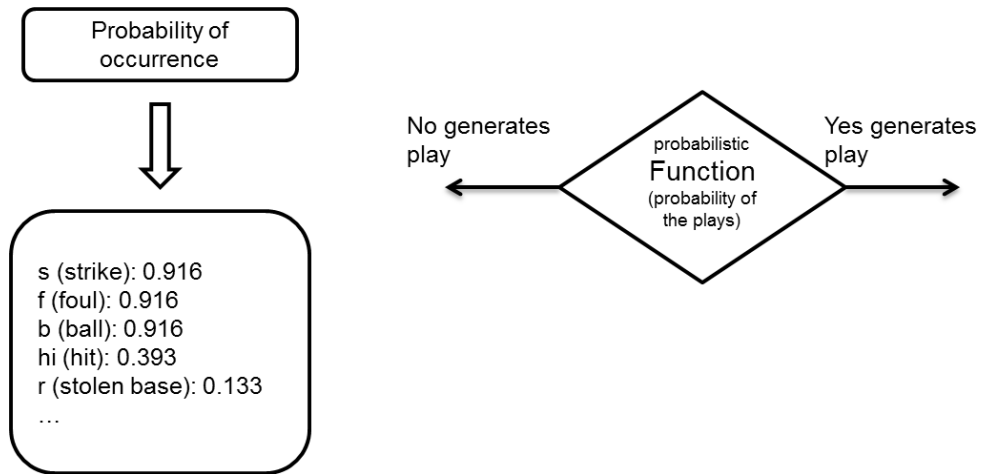


Figure 2.4 Probabilistic function scheme.

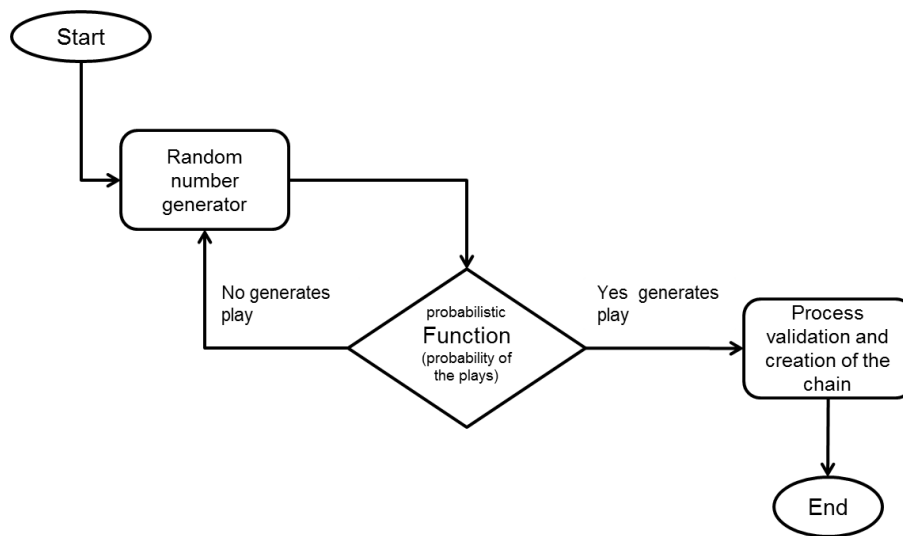


Figure 2.5 General scheme of the generation and construction of chains.

Table 2.6 Algorithm for baseball plays generation.

-
- Step 1: Numbers in $\{0, \dots, m\}$ are random generated and each one is associated to any of the m simplest Baseball plays.
 - Step 2: String of play is created, including the sequence of prior single plays if needed.
 - Step 3: Validation of the string as a Baseball play.
 - Step 4: After get a valid play, a flip probabilistic function is used to decide to execute the play or not up to its SO.
-

Step 5: If Baseball match simulation is over go to Step 6, otherwise go to Step 1.
Step 6: End of simulation: the validated string is a part of or a whole match.

2.5 Selection of Strategies and Payoff Matrices

In this section we define some mathematical aspects for the strategic analysis using NE and PE.

The NE and PE formal account for multi-player games follow. The joint actions from all the players set the strategy profiles vectors; position i corresponds to the player i action. Let $P = \{1, \dots, n\}$ be the set of players, $i \in P$, $a_x^i \in \Sigma^i$ be an element of the set of simple plays, and s_x^i be a strategy of player i , $s_x^i \in S_i$; let $G = (S_1, \dots, S_n; u_1, \dots, u_n)$ be the game in normal form [81] where:

- A strategy is a compound sequence $s_x^i = a_1^i \dots a_n^i$.
- A strategy profiles is an n -tuple of strategies one strategy per player (s_1, \dots, s_n) .
- S_i is the set of strategies of player i .
- $\{S_1, \dots, S_n\}$ is the set of all the S_i strategies.
- $\{u_1, \dots, u_n\}$ is the set of all payoff functions one per player.

2.5.1 Nash Equilibrium for Non-Cooperation Strategies

The NE for Baseball strategic reasoning induces each player's action-decision during a match by regarding the others players' action-decision, hence the team's action-decision should fit the NE strategy profile being modeled by the payoff matrices for a quantitative analysis.

The basic concepts to fundament NE follow. During a game the strategy profile specifies the whole team's actions being derived from every player's strategy. A *strategy profile is dominated* if any player can be benefit improved by deviating to other of his/her strategies into a different profile. A *deviation strategy profile* is a set of deviation profiles. A *payoff function* is for calculating the benefit obtained for every possible strategy profile in the game [81].

The Nash equilibrium [81] is a widely used mathematical concept in game theory, especially in non-cooperative games. To identify the strategy profiles that satisfy the condition of Nash equilibrium, every strategy profile is evaluated with the payoff functions of the players, and the chosen profiles are those which for every player be the options that produces less loss for them, individually, non-cooperative, the best options for each player. The mathematical definition is given below.

The NE strategies are denoted s_1^*, \dots, s_n^* and s_i^* is the best answers from player i to the $n - 1$ other players' strategies, $s_1^*, \dots, s_{i-1}^*, s_{i+1}^*, \dots, s_n^*$; $(s_1^*, \dots, s_i^*, \dots, s_n^*)$ is the n -tuple of strategies for payoff function maximum equation (2.1):

$$u_i(s_1^*, \dots, s_{i-1}^*, s_i^*, s_{i+1}^*, \dots, s_n^*) \geq u_i(s_1^*, \dots, s_{i-1}^*, s_i, s_{i+1}^*, \dots, s_n^*) \quad \forall i \in P, s_i \in S_i \quad (2.1)$$

Every strategy profile is each payoff function valued and compared with all of the others to determine whether it is or is not dominated. The dominated profiles are discarded and the non-dominated profiles fit the NE (see Table 2.7). Any game in (finite) normal form at least has one strategy profile that fits the NE [81].

Table 2.7 Nash equilibrium algorithm for selection of strategies in baseball gaming.

| |
|---|
| Input each strategy profile and its payoff value |
| 1: for all $sp = (sp_1, \dots, sp_m)$ strategy profiles |
| 2: for all player $i = (1, \dots, n)$ |
| 3: if sp is labelled as non-dominated |
| 4: Do the derivations in sp for player i |
| 5: if sp is dominated by at least one derivation of i |
| 6: labeled sp as dominated, move to the next strategy profile |
| 7: end if |
| 8: end if |
| 9: else move to the next strategy profile |
| 10: end for |
| 11: end for |

Observe that in NE every player is applying a non-cooperative perspective – less bad for him regarding the other players' strategies.

2.5.2 Pareto Efficiency for Cooperation Strategies

In a broad perspective to deal with valuations on strategy profiles for multiple players, definition of Pareto dominance follows: a vector $\vec{v} = (v_1, \dots, v_k)$ is said to dominate $\bar{v} = (\bar{v}_1, \dots, \bar{v}_k)$ if and only if \vec{v} is at least partially better off than \bar{v} , formally in (2.2) [74].

$$\forall j \in \{1, \dots, k\}, v_j \geq \bar{v}_j \wedge \exists j' \in \{1, \dots, k\}: v_{j'} > \bar{v}_{j'} \quad (2.2)$$

Let $x = (s_1, \dots, s_n)$ be a strategy profile, and $\vec{u} = (u_1(x), \dots, u_n(x))$ be the vector with all of the valuations from payoff functions $u_i, i \in P$. Vector \vec{u} is Pareto efficient (PE) if and only if there is not another vector \bar{u} which dominates \vec{u} . Thus, one strategy profile results in a Pareto efficient valuation if and only if it is not dominated. In other words, a strategy profile is Pareto efficient valued if there is no other strategy profile such that all players are better off and at least one player is strictly better off. Algorithm for PE is in Table 2.8.

Table 2.8 Pareto efficiency algorithm for selection of strategies in baseball gaming.

Input each strategy profile and its payoff value

1: for each $x = (x_1, \dots, x_n)$ strategy profiles

2: Create the vectors $u(j) = (u_1(x), \dots, u_n(x)), j = 1$ to the total number of profiles

3: end for

4: $PF = \text{find} - \text{nondominate}(v)$, PF contains profiles which are Pareto efficient

5: find in PF the profile(s) which is (are) cooperative (for all players as team)

Pareto efficiency (PE) or optimality is foundational for comparisons and discussions on social welfare and choice, as well as on the use of social welfare functions [88]. By applying Pareto efficiency for selection of strategy profiles in Baseball, we select those where the profits are maximized as a group and not only individually. Each player uses the strategy such that all players, as a team, get maximum utility, so multi-player cooperation is achieved. In this thesis, from a set of Pareto efficient profiles, we select those

where the profits are maximized as a group and not only individually i.e., each player uses the strategy such that all players get maximum utility cooperatively.

The FSM that models strategies profiles is as follows. Let $(\Sigma, \hat{S}, s, \varphi, H)$ be the FSM for modeling the NE or Pareto efficient strategies profiles of all player in a team, see Figure 2.6.

- $\Sigma = \{a_1, a_2, \dots, a_n, \varepsilon, \theta\}$ is the alphabet.
- $\hat{S} = \{s, s_0, \dots, s_n, h\}$ is the set of states.
- $\varphi: \hat{S} \times \Sigma \rightarrow \hat{S}$ is the transitions function.
- $s \in \hat{S}$ is the initial state.
- $H = \{h\} \subseteq \hat{S}$ is the set of halt states.

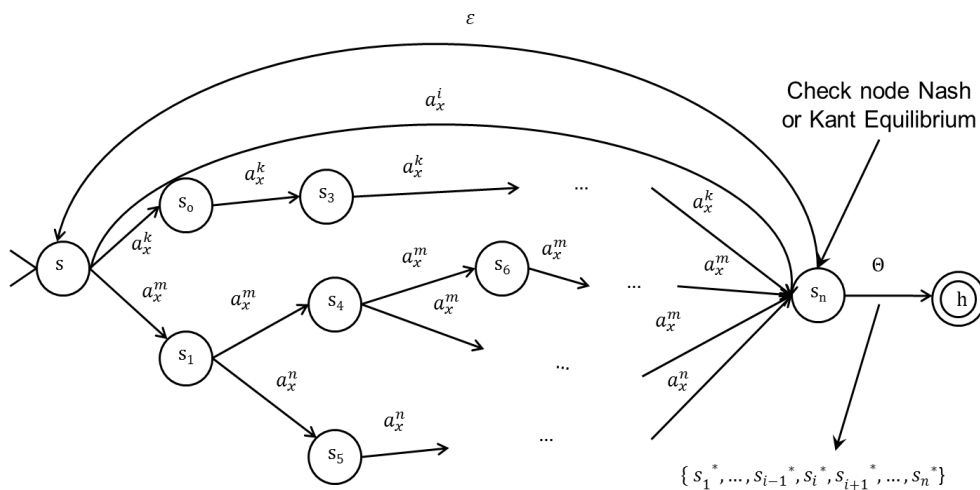


Figure 2.6 FSM for modeling the Nash or Pareto efficient strategies profiles of a team.

2.5.3 Modeling of Payoff Matrices

Matrices entries registry every player's decisions to shape the team's strategy profiles, in addition to the match score and the number of innings and outs, see Figure 2.7. Every strategy profile is analyzed and those that do not fit the NE or PE condition are discarded.

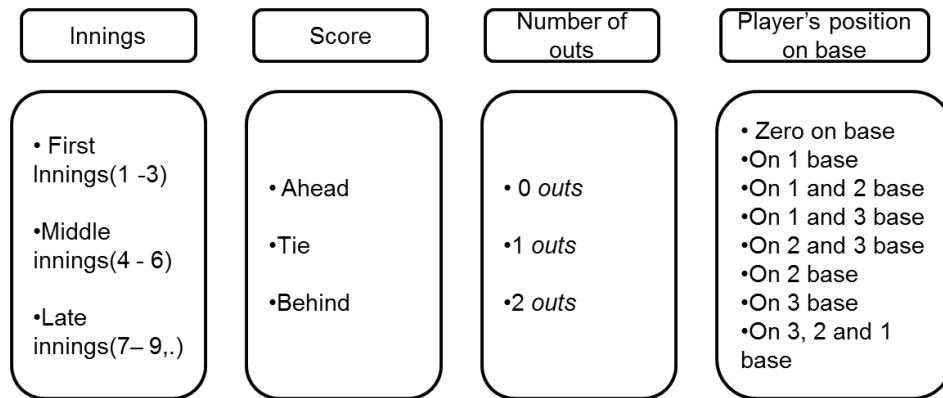


Figure 2.7 Matrix entries for a baseball quantitative analysis.

From the set of M^i payoff matrix of every player i the M payoff matrix for n players is arranged. The M entries are the strategy profiles joint to the profile payoff value r_z , hence the form of a matrix entry is $((s_1, \dots, s_n), r_z)$. An example of two-player's profiles with three strategies per player is present in Figure 2.8, and the payoff matrix for one of the players is in Figure 2.9.

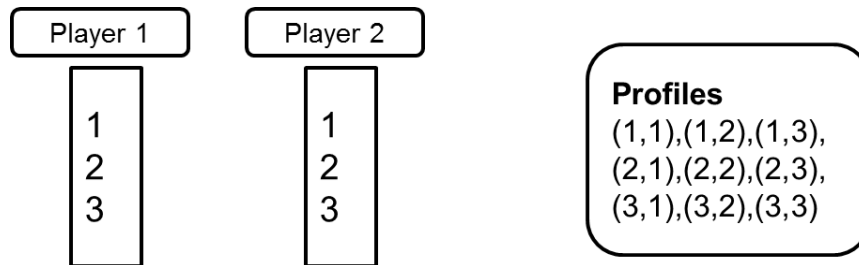


Figure 2.8 Two players' strategy profiles.

| | |
|-------|-------|
| (1,1) | r_1 |
| (1,2) | r_2 |
| (1,3) | r_3 |
| (2,1) | r_4 |
| (2,2) | r_5 |
| (2,3) | r_6 |
| (3,1) | r_7 |
| (3,2) | r_8 |
| (3,3) | r_9 |

(s_x^1, s_y^2) are the game profiles, and where r_z is the payoff value

Figure 2.9 Payoff matrix for one player in a two-player game.

The quantitative analysis for a whole Baseball match is based on a set of 216 information files, which are the combinations from, 3 stages of the game (first, middle and late innings), 3 score conditions, 3 amounts of outs (0 or 1 or 2) and 8 player's position on base, each file contains 1 to 4 payoff matrices up to the match circumstances being represented.

A remark is that in a Baseball match, at any moment, any change of players is allowed, what is Baseball characteristic not for several field games. This manager's decision making alternative can be benefitted from the NE or Pareto efficient strategy profile guidance as long as the best combination of players according to a strategy at any moment of the match is ever available.

2.5.3.1 Players' Payoff Functions

In this section, payoff functions for the Baseball runners and batters are defined. Parameter δ indicates the score conditions, $\delta = 1$, $\delta = 0.5$ and $\delta = 0.2$ when team is down, tied and up on score. Parameter η gives information about innings: $\eta = 0.2$, $\eta = 0.5$ and $\eta = 1$ when the match is in first (1-3), middle (4-6) and late (7-9 or extra innings) innings. Parameter β gives information about the number of outs in the inning: $\beta = 0$ for two outs and $\beta = 1$ for one or zero outs.

The batter is identified with a and the runners with b, c, d . Let $\psi \in [0, 1]$ be a weighting factor used in a runners' payoff function to consider the batter's strategies; and let $\gamma, \mu \in [0, 0.1]$ be parameters to determine the playing style by regarding in turns parameters α and η . Let ρ_{si} be the statistical occurrence (SO) of the strategy s for player i and v_i is the preference value of the player i to the profile that is being analyzed. The payoff function is given by ρ_{si} and the parameters described previously.

The strategy profiles as $(s_1, \dots, \mathbf{s}, \dots, s_n)$ highlighting in bold the focus player's strategy. We observe that Baseball strategy profiles for the offensive team, with men in the bat position and runners are at most a 4-tuple; actually, 3 runners at most in the field, one per base, and the batter. Hence strategy profile analysis is restricted by this condition. To define payoff functions, we should regard combinations according to the next conditions.

Payoff function for the runners

Let $u_b(s_1, \dots, \mathbf{s}, \dots, s_n)$ be the runner b 's payoff function, with $s \in \{r, wb\}$; r notation is for try to steal the forward base, and wb for wait the batter's action.

Case man on 1st or 2nd base

If runner's SO to steal a base ρ_{rb} is such that, $\rho_{rb} < 1 - \rho_{rb}$, payoff function is in equation (2.3),

$$u_b(s_1, \dots, \mathbf{r}, \dots, s_n) = 0 + v_b, u_b(s_1, \dots, \mathbf{wb}, \dots, s_n) = 0.1 + v_b. \quad (2.3)$$

Otherwise, if $\rho_{rb} > 1 - \rho_{rb}$, we need to consider if any SO of batter' strategy is greater than ψ , and use equation (2.4),

$$u_b(s_1, \dots, \mathbf{r}, \dots, s_n) = \psi * \beta + v_b, u_b(s_1, \dots, \mathbf{wb}, \dots, s_n) = \psi + v_b. \quad (2.4)$$

Otherwise, use equation (2.5),

$$u_b(s_1, \dots, \mathbf{r}, \dots, s_n) = 0.1 + v_b, u_b(s_1, \dots, \mathbf{wb}, \dots, s_n) = 0 + v_b. \quad (2.5)$$

Case men 2nd and 1st

Let runner b the man on 2nd base and runner c the man on 1st base. In this case the advance of c depends on the advance of b , which payoff function is obtained as in the previous case. If ρ_{rc} steal a base from c is such that $\rho_{rc} < 1 - \rho_{rc}$, use equation (2.6),

$$u_c(s_1, \dots, \mathbf{r}, \dots, s_n) = 0 + v_c, u_c(s_1, \dots, \mathbf{wb}, \dots, s_n) = 0.1 + v_c. \quad (2.6)$$

Otherwise, if $\rho_{rc} > 1 - \rho_{rc}$, consider the decision of runner b , if $u_b(s_1, \dots, \mathbf{r}, \dots, s_n) > u_b(s_1, \dots, \mathbf{wb}, \dots, s_n)$, so if runner b tries to steal base greater than wait for the batter's action, and use equation (2.7),

$$u_c(s_1, \dots, \mathbf{r}, \dots, s_n) = \rho_{rc} * \beta + v_c, u_c(s_1, \dots, \mathbf{wb}, \dots, s_n) = \rho_{rc} + v_c. \quad (2.7)$$

Otherwise, use equation (2.8),

$$u_c(s_1, \dots, \mathbf{r}, \dots, s_n) = 0 + v_c, u_c(s_1, \dots, \mathbf{wb}, \dots, s_n) = 0.1 + v_c. \quad (2.8)$$

Case men on 3rd, or 3rd and 2nd, or 3rd, 2nd and 1st

In this case, the payoff function may include when b is in 3rd base, c in 2nd base and d in 1st base. Base stealing is neutralized since it is highly unlikely that any runner tries base stealing in these positions, and use equations in (2.9).

$$\begin{aligned} u_b(s_1, \dots, \mathbf{r}, \dots, s_n) &= 0 + v_b, u_b(s_1, \dots, \mathbf{wb}, \dots, s_n) = 0.1 + v_b. \\ u_c(s_1, \dots, \mathbf{r}, \dots, s_n) &= 0 + v_c, u_c(s_1, \dots, \mathbf{wb}, \dots, s_n) = 0.1 + v_c. \\ u_d(s_1, \dots, \mathbf{r}, \dots, s_n) &= 0 + v_d, u_d(s_1, \dots, \mathbf{wb}, \dots, s_n) = 0.1 + v_d. \end{aligned} \quad (2.9)$$

Payoff function for the batter

To explain To explain how to define the payoff function for batters, we use the strategies, home run h , hit hi , sacrifice flies fs and sacrifice bunt tb , so, $s \in \{h, hi, fs, tb\}$,

even other strategies may be also used. The γ , μ parameters indicate the playing style, aggressive or conservative, according to results of a set of experiments, and by regarding to the score condition α and the information on innings, η . To model an aggressive style use ($\delta * \eta == 1$), that yield to $\mu = 0.03, \gamma = 0$. For a conservative style use ($\delta * \eta == 0.5$) that yields to $\mu = 0, \gamma = 0.08$. Otherwise, $\mu = 0, \gamma = 0$ that means that these parameters do not affect the playing style, and playing is sole restricted to characteristic of players.

The values of μ and γ came from some experiments, and their assignation is open. They are independent and will affect to different Baseball plays in order to induce the playing style. They may be asymmetrical or not. The value of μ will weight to home run and hit plays, for playing aggressively, and the value of γ will weight to sacrifice fly, for playing conservatively

Case: no-runner

With no runners on bases, we only consider the playing style and the SO of batter' strategy to define the payoff function in equation (2.10).

$$\begin{aligned}
 u_a(\mathbf{fs}) &= 0. \\
 u_a(\mathbf{tb}) &= p_{tba} * \beta + \gamma. \\
 u_a(\mathbf{h}) &= \rho_{ha} + \mu. \\
 u_a(\mathbf{hi}) &= \rho_{hia} + \mu.
 \end{aligned} \tag{2.10}$$

Case: one man on base

Runner b is the man in base. In this case consider the statistical occurrence of batter a strategies, the preference value v_a of the batter a on strategy profile (s, s_1) , $s_1 \in \{r, wb\}$, the statistical occurrence ρ_{s_1b} from runner b on strategy s_1 , and the playing style. The payoff function follows in equation (2.11).

$$\begin{aligned}
 u_a(\mathbf{fs}, s_1) &= ((\rho_{fsa} + v_a) - \rho_{s_1b}) * \beta + \gamma. \\
 u_a(\mathbf{tb}, s_1) &= ((\rho_{tba} + v_a) - \rho_{s_1b}) * \beta + \gamma. \\
 u_a(\mathbf{h}, s_1) &= ((\rho_{ha} + v_a) - \rho_{s_1b}) + \mu.
 \end{aligned} \tag{2.11}$$

$$u_a(\mathbf{hi}, s_1) = ((\rho_{hia} + v_a) - \rho_{s_1b}) + \mu.$$

Case: two men on base

Runner b is the more advanced runner in base and c is the other runner. In this case, we consider the statistical occurrence of each batter a strategies, the preference value v_a of the batter a on strategy profile (s, s_1, s_2) , $s_1, s_2 \in \{r, wb\}$ for the runners b and c , the statistical occurrence of strategies s_1 and s_2 from b and c for the, respectively, and the playing style. The payoff function follows in equation (2.12).

$$\begin{aligned} u_a(\mathbf{fs}, s_1, s_2) &= ((\rho_{f sa} + v_a) - (\rho_{s_1b} + \rho_{s_2c})) * \beta + \gamma. \\ u_a(\mathbf{tb}, s_1, s_2) &= ((\rho_{t ba} + v_a) - (\rho_{s_1b} + \rho_{s_2c})) * \beta + \gamma. \\ u_a(\mathbf{h}, s_1, s_2) &= ((\rho_{ha} + v_a) - (\rho_{s_1b} + \rho_{s_2c})) + \mu. \\ u_a(\mathbf{hi}, s_1, s_2) &= ((\rho_{hia} + v_a) - (\rho_{s_1b} + \rho_{s_2c})) + \mu. \end{aligned} \tag{2.12}$$

Case: there men on base

In this case, consider the statistical occurrence of each batter a strategies, the preference value v_a of the batter a on strategy profile (s, s_1, s_2, s_3) , $s_1, s_2, s_3 \in \{r, w\}$ for the runners b, c and d , the statistical occurrence of the strategies s_1, s_2 and s_3 from b, c and d , respectively, and the playing style. The payoff function follows in equation (2.13).

$$\begin{aligned} u_a(\mathbf{fs}, s_1, s_2, s_3) &= ((\rho_{f si} + v_a) - (\rho_{s_1b} + \rho_{s_2c} + \rho_{s_3d})) * \beta + \gamma. \\ u_a(\mathbf{tb}, s_1, s_2, s_3) &= ((\rho_{t ba} + v_a) - (\rho_{s_1b} + \rho_{s_2c} + \rho_{s_3d})) * \beta + \gamma. \\ u_a(\mathbf{h}, s_1, s_2, s_3) &= ((\rho_{ha} + v_a) - (\rho_{s_1b} + \rho_{s_2c} + \rho_{s_3d})) + \mu. \\ u_a(\mathbf{hi}, s_1, s_2, s_3) &= ((\rho_{hia} + v_a) - (\rho_{s_1b} + \rho_{s_2c} + \rho_{s_3d})) + \mu. \end{aligned} \tag{2.13}$$

2.5.3.2 Examples of Analysis of Strategies

Consider the followings circumstances in a Baseball match: last innings with the match score tied, one out in the inning and runner b in 3rd base. The b options are, base steal-

ing r , or wait (wb) for the batter's action. For batter a options are, *homerun* h , or a *sacrifice hit* (fs). Let $\beta = 1$, $\delta = 1$ and $\eta = 0.5$, hence $\mu = 0$, and $\gamma = 0.08$, so playing style is conservative. Using payoff functions the strategy profiles values are calculated to identify the ones that fit NE or PE.

For runner b , let his SO of $\rho_{rb} = 0.2$. Using the payoff function (2.3).

- $u_b(h, r) = 0 + v_b = 0 + 0.5 = 0.5$. $u_b(h, wb) = 0.1 + v_b = 0.1 + 0.4 = 0.5$.
- $u_b(fs, r) = 0 + v_b = 0 + 0.3 = 0.3$. $u_b(fs, wb) = 0.1 + v_b = 0.1 + 0.3 = 0.4$.

For batter a , using payoff function (2.11).

- $u_a(h, r) = ((\rho_{ha} + v_a) - \rho_{rb}) + \mu = ((0.3 + 0.4) - 0.2) + 0 = 0.5$.
- $u_a(h, wb) = ((\rho_{ha} + v_a) - \rho_{wbb}) + \mu = ((0.3 + 0.4) - 0.8) + 0 = -0.1$.
- $u_a(fs, r) = ((\rho_{fsa} + v_a) - \rho_{rb}) * \beta + \delta = ((0.62 + 0.0) - 0.2) * 1 + 0.08 = 0.5$.
- $u_a(fs, wb) = ((\rho_{fsa} + v_a) - \rho_{wbb}) * \beta + \delta = ((0.62 + 0.4) - 0.8) * 1 + 0.08 = 0.3$.

The strategy profiles and the utility value assigned by the payoff function of each player to each profile are shown in Fig. 2.10.

| | $x = (s_1, \dots, s_n)$ | $u_a(x)$ | $x = (s_1, \dots, s_n)$ | $u_b(x)$ |
|---|-------------------------|----------|-------------------------|----------|
| ① | (h, r) | 0.5 | (h, r) | 0.5 |
| ② | (h, wb) | -0.1 | (h, wb) | 0.5 |
| ③ | (fs, r) | 0.5 | (fs, r) | 0.3 |
| ④ | (fs, wb) | 0.3 | (fs, wb) | 0.4 |

Figure 2.10 Entries for the payoff matrices of players 1 and 2

Now, in Fig. 2.11 the deviations in the strategy profiles is illustrated, such that in the analysis, depending on the values assigned by the payoff function, those strategy profiles being not dominated are identified. The example illustrates the steps to be applied to find the profiles that satisfy NE condition. For a player, x_1/x_2 means that profile x_1 dominates profile x_2 , so for player 1 we have 2/4; for player b domination is by 3/4.

Therefore, the non-dominated profiles for all players are the profiles 1, (h, r) and 4, (fs, wb) , and both satisfy the NE condition. The only profile that satisfies PE condition is (h, r) because, in this profile, both players get the maximum profit as a team.

| | $x = (s_1, \dots, s_n)$ | $u_a(x)$ | | $x = (s_1, \dots, s_n)$ | $u_b(x)$ |
|---|-------------------------|----------|---|-------------------------|----------|
| | (h, r) | 0.5 | | (h, r) | 0.5 |
| ② | (h, wb) | -0.1 | | (h, wb) | 0.5 |
| | (fs, r) | 0.5 | ③ | (fs, r) | 0.3 |
| ④ | (fs, wb) | 0.3 | ④ | (fs, wb) | 0.4 |

Figure 2.11 Deviations in the strategy profiles

Table 2.9 summarizes: the strategy profiles, the payoff values assigned to profiles by each player, the statistical occurrence of strategies, and the profiles which are NE or Pareto efficient or none, following the example above. Usually, the strategies in NE profiles are statistically more frequent of occurrence than strategies in Pareto efficient profiles. Particularly, sacrifice hit (fs) strategy is in NE profile and home run (h) is in Pareto efficient profile; statistically, fs is more frequent to occur than h , although h is more profitable than hi . The Pareto efficient profiles are the theoretical most profitable, but their occurrence in practice is too low.

Table 2.9 Summary of the analysis of strategy profiles

| | Strategy profiles | Payoff value by player | Statistical occurrence (average) | NE | PE |
|---|-------------------|------------------------|----------------------------------|----|----|
| 1 | (h, r) | 0.5, 0.5 | 0.3, 0.2 | ✓ | ✓ |
| 2 | (h, wb) | -0.1, 0.5 | 0.3, 0.8 | | |
| 3 | (fs, r) | 0.5, 0.3 | 0.7, 0.2 | | |
| 4 | (fs, wb) | 0.3, 0.4 | 0.7, 0.8 | ✓ | |

Profiles (h, r) and (fs, wb) fit the NE condition because hold equation (2.1)

- $u_a(h, r) \geq u_a(fs, r)$
- $u_b(h, r) \geq u_b(h, wb)$

and

- $u_a(fs, wb) \geq u_a(h, wb)$
- $u_b(fs, wb) \geq u_b(fs, r)$

Profile (h, r) is Pareto efficient $\vec{u} = ((u_a(h, r), u_b(h, r)))$ because given $s_1 \in \{hi, h\}, s_2 \in \{r, wb\}$, there is not a vector $\vec{u} = (u_a(s_1, s_2), u_b(s_1, s_2))$ that dominates \vec{u} , see equation (2.2).

Next, the analytical comparison of the use of HM for player-positions assignment combined with NE or PE for selection of strategies follows. The way to assign player-positions is of great importance and a team perspective analysis is needed. In addition, the selection of strategies is essential for a good team performance.

2.6 Hungarian Algorithm for Baseball Team Selection

The Baseball players' selection has as primordial goal to attain overall team efficiency. A baseball team must not be formed based on individual skills of its players; it needs to consider the contribution of each one on his assigned position for the best team performance. To deal with the problem of Baseball positions assignment, we used Hungarian algorithm proposed by Kuhn [66] and Munkres [79], statistic of Major League Baseball (MLB) as the data set and the methodology is based on the proposal of Britz and Maltitz [18]. The teams comparative analysis using different techniques was performed in a Baseball simulator [3].

Hungarian algorithm (HA) is a prime method to solve the assignment problem [66]. HA is been using in different problems such as: Baseball players' selection, economic issues, clustering analysis.

Given the necessity of select the most effective Baseball team, a set of measures to assess the players' skills is required. Britz and Maltitz [18] proposed a set of tests, in order

to evaluate each player to know his skills. Our approach uses Britz and Maltitz's methodology and real statistics of MLB players.

The information source used to obtain statistics of players was the data by seasons of MLB Baseball teams. Precisely, we used 42 players for the 2012 MLB season, including 21 of the team of Boston Red Sox and 21 of New York Yankees. From these players, we selected 12 pitchers (6 from each team) and 30 field players (15 from each team). Statistical measures considered are shown in Table 2.10:

Table 2.10 Statistical measures to be considered.

| Batting | | Pitching | | Fielding | |
|----------------|-----------------|-----------------|-------------------------|-----------------|--------------------|
| R | Runs | W-L% | Percentage of game won | E | Errors |
| SB | Stealing bases | ERA | Runs allowed per game | DP | Double plays |
| OPS | OBP + SLG | WHIP | (BB + Hits) / Inning | RF/9 | (PO + A) / Innings |
| GDP | Double plays | H/9 | Hits per game | SB | Stealing bases |
| SH | Sacrifices hits | HR/9 | Home run per game | | |
| SF | Sacrifices fly | BB/9 | Bases on balls per game | | |
| IBB | BB intentional | SO/9 | Strikeout per game | | |

**OBP percentage of times an offensive player reaches a base.*

**SLG total bases reached by total number at-bat.*

**BB bases on balls allowed the player.*

**PO outs got for the defensive.*

**A assistance to get outs.*

2.6.1 Data Normalization

The statistical data must be normalized, such that, it does not affect comparisons. The following expressions were taken from [18], the normalization process is as follows.

Let l_i, u_i be the lowest and highest value, respectively, of a measure $i = 1 \dots n$. The relative score (RS) for each observation j in test i is a transformation of the absolute value (AS) as noted as follows (2.14):

$$RS_{ij} = \frac{(AS_{ij}-l_i)}{(u_i-l_i)}, \quad j = 1 \dots n \quad (2.14)$$

Once all the tests score have been converted, this information can be summarized in the following matrix (2.15):

$$T = \begin{pmatrix} RS_{11} & \dots & RS_{1t} \\ \vdots & \ddots & \vdots \\ RS_{n1} & \dots & RS_{nt} \end{pmatrix} \quad (2.15)$$

Where t is the number of tests and n is the number of players. The purpose of the skills tests is to correlate each player to a Baseball position, for this, we define a weight vector for each position such that ponders its relationship with each test. The weight vector for position i is represented as $w^i = (w_1^i, w_2^i, \dots, w_t^i)$, $w_j^i \geq 0$, $\sum_{j=1}^t w_j^i = 1$. Each w_j^i represents the importance of test j corresponding to the position i . These vectors comprise the matrix W such that $w_j^i = w_{ji}$ are the entries of the matrix (2.16):

$$W = \begin{pmatrix} w_{11} & \dots & w_{1k} \\ \vdots & \ddots & \vdots \\ w_{t1} & \dots & w_{tk} \end{pmatrix} \quad (2.16)$$

Where k is the number of positions and t is the number of tests. The selection of weight vectors is the responsibility of an expert and should be selected carefully, with the matrices T and W , we obtain the relationship between each player and each position that is given by the cost matrix C (2.17):

$$C = T * W = \begin{pmatrix} RS_{11} & \dots & RS_{1t} \\ \vdots & \ddots & \vdots \\ RS_{n1} & \dots & RS_{nt} \end{pmatrix} * \begin{pmatrix} w_{11} & \dots & w_{1k} \\ \vdots & \ddots & \vdots \\ w_{t1} & \dots & w_{tk} \end{pmatrix} = \begin{pmatrix} q_{11} & \dots & q_{1k} \\ \vdots & \ddots & \vdots \\ q_{n1} & \dots & q_{nk} \end{pmatrix} \quad (2.17)$$

The solution to our problem is by finding the combination of values in C such that maximizes the efficiency subject to certain constraints:

1. Select exactly one value for each column, to ensure that each position is assigned to a player.
2. Select at most one value for each row, to ensure that a player is no assigned to more than one position.

Stated mathematically, the optimal team efficiency is defined as (2.18):

$$E = \max_Y Z(Y) = \sum_{i=1}^n \sum_{j=1}^k y_{ij} q_{ij} \quad (2.18)$$

Where $y_{ij} = \{0, 1\}$, subject to:

$$\sum_{i=1}^n y_{ij} = 1 \quad , \quad j = 1 \dots k$$

$$\sum_{j=1}^k y_{ij} \leq 1 \quad , \quad i = 1 \dots n$$

The objective function Z represents the overall efficiency of the team and the combination of player yielding E is the optimal selection of the team. The player i is assigned to the position j if $y_{ij} = 1$. Note that the simple selection of the fittest players for each position does not ensure the optimal selection. The solution is more complex than that and to find it, Hungarian algorithm is used.

2.6.2 Hungarian Algorithm

Unlike the initial proposal by Kuhn, in this case the assignment is not complete, i.e., there are more players than positions. In addition, this is an efficiency maximization problem, is not a minimization problem. The algorithm to solve the problem follows.

Given the matrix that contains the contributions of a set of agents (rows) to a set of tasks (columns), the following procedure ensures an optimal assignment:

Step 1: Convert the values in the matrix of maximum benefit at lower cost. This is done by subtracting each element, the maximum value of the matrix (2.19):

$$c_{ij} = M - c_{ij}, \text{ where } M = \max_{\substack{i=1\dots n \\ j=1\dots k}} c_{ij} \quad (2.19)$$

Step 2: If the matrix is a rectangular, transform it to a square matrix, adding columns missing and assigning to these position a value greater than M .

Step 3: In each row, subtract to each element the minimum row value.

Step 4: For each column, subtract to each element the minimum column value.

These two steps ensure that there will be at least one zero in each row and column.

Paso 5: Draw lines through rows and columns in such way that all zeros are covered using the minimum number of lines. Let k be the number of lines used.

a) If $k = n$ go to step 6.

b) If $k < n$, let m be the smallest number that is not covered by any lines. Every element not covered for any lines, subtract m (including m). Add m to every number in a position where two lines intersect. Repeat the step 5 until $k = n$.

Step 6: Evaluate each row, starting at the top and highlight the zeros which are the solitary zeros in the row. The positions of these zeros are unique assignments and, therefore, the corresponding row and column can be deleted from further consideration. If all n assignments have not been made by applying this step, repeat this procedure for the columns, starting from the left. Continue iterating between rows and columns until all n assignments have been made. If a final complete solution cannot be reached this means

that there is no unique solution yielding the minimum overall cost. An arbitrary zero can then be selected and Step 6 can be repeated if necessary for the remaining rows and columns, producing a final assignment solution. At the end of this algorithm, all highlighted zeros in positions q_{ij} means assign player i to position j .

2.7 Merging Equilibrium of Nash and Pareto Efficiency: Test and Comparison

We do an analytical comparison on the teams gaming performance. Experiments concern the performance comparison of teams that use a method for selection of strategies with regard to the next match gaming conditions:

- Comparing the MLB results from some teams against the simulation results by applying NE or PE.
- A team with a score disadvantages changes from NE to PE, and vice-versa (PE to NE).
- Using PE by a defense team for exclusive, versus the NE use by offensive teams for exclusive.

2.7.1 Simulation using MLB Data or Selection of Strategies

To simulate the players' actions according to their performance, we use MLB real statistics from the New York Yankees (NYY) and Oakland Athletics (OAK) in the 2012 season (some data are in Table 2.11). Shown is the number of times that a player makes **AB** at bat, **R** for reach home base, **H** for a hit, **2B** for a hit and reaches second base, **3B** for a hit and reach third base, **HR** for a home run, **BB** for walk by a player (four balls during at bat), **SO** for strikeout (three strikes during at bat), **SB** for stolen a base, **CS** for a player put out by attempting to steal a base.

Table 2.11 Some MLB baseball players' statistics.

| Player | Team | AB | H | 2B | 3B | HR | BB | SO | SB | CS |
|-------------|------|-----|-----|----|----|----|----|-----|----|----|
| Suzuki, I | NYY | 227 | 73 | 13 | 1 | 5 | 5 | 21 | 14 | 5 |
| Jeter, D | NYY | 683 | 216 | 32 | 0 | 15 | 45 | 90 | 9 | 4 |
| Cano, R | NYY | 527 | 196 | 48 | 1 | 33 | 61 | 96 | 3 | 2 |
| Nunez, E | NYY | 89 | 26 | 4 | 1 | 1 | 6 | 12 | 11 | 2 |
| Chavez, E | NYY | 278 | 78 | 12 | 0 | 16 | 30 | 59 | 0 | 0 |
| Swisher, N | NYY | 537 | 146 | 36 | 0 | 24 | 77 | 141 | 2 | 3 |
| Cespedes, Y | OAK | 487 | 142 | 25 | 5 | 23 | 43 | 102 | 16 | 4 |
| Moss, B | OAK | 265 | 77 | 18 | 0 | 21 | 26 | 90 | 1 | 1 |
| Gomes, J | OAK | 279 | 73 | 10 | 0 | 18 | 44 | 104 | 3 | 1 |
| Crisp, C | OAK | 455 | 118 | 25 | 7 | 11 | 45 | 64 | 39 | 4 |
| Reddick, J | OAK | 611 | 148 | 29 | 5 | 32 | 55 | 151 | 11 | 1 |
| Smith, S | OAK | 383 | 92 | 23 | 2 | 14 | 50 | 98 | 2 | 2 |

Using the MLB statistics [77], the frequency of occurrence of each Baseball play per player is used to induce the SO of the play can happen in a match, e.g., the SO of a player making a hit is given by $\mathbf{AB/H}$, a home run by $\mathbf{AB/HR}$ and so on. Thus, when a player is at bat, we can simulate his performance in a gaming (e.g. 2012) season. Next, we do a comparison among simulations of Baseball matches using MLB statistics, without any concern for analysis of strategies, versus simulations that use NE or PE as the methods for selection of strategies. Two hundred computer simulations per each of the next conditions were carried out.

- 1) Team 1 (T_1) uses NE versus Team 2 (T_2) uses NYY MLB statistics.
- 2) T_1 uses NE versus T_2 uses OAK MLB statistics.
- 3) T_1 uses PE versus T_2 uses NYY MLB statistics.
- 4) T_1 uses PE versus T_2 uses OAK MLB statistics.

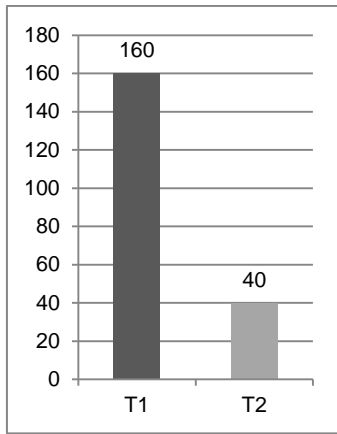


Figure 2.12 T₁ NE vs T₂ NYY.

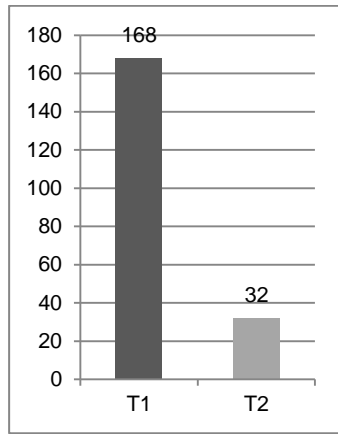


Figure 2.13 T₁ NE vs T₂ OAK.

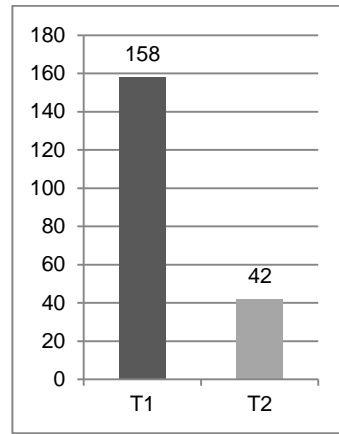


Figure 2.14 T₁ PE vs T₂ NYY.

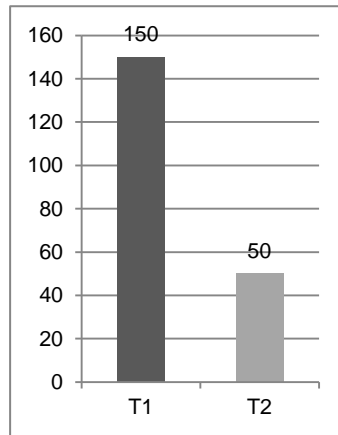


Figure 2.15 T₁ PE vs T₂ OAK.

By considering the results in Figure 2.12, when T₁ uses NE and T₂ uses NYY statistics, T₁ is 160/40 superior. Results in Figure 2.13 show when T₁ uses NE and T₂ uses OAK statistics, and T₁ is 168/32 superior. Results in Figure 2.14 show when T₁ uses PE and T₂ uses NYY statistics, and 158/42 wins in favour of T₁. Results in Figure 2.15 illustrate when T₁ uses PE and T₂ uses OAK statistics, and T₁ won more times 150/50.

The huge contrast between the results from the previous simulations quantifies the relevance of the selection of strategies, even for a team having top level Baseball players, whose inclusion does not guarantee a high level team performance. Therefore, methods

analysis for guiding players' actions as a team is primordial for the selection of the proper strategies to increase the probability of team success in a match.

2.7.2 The Mix Election Methods

Two hundred simulations per each of the next selection of strategies, sometimes combinations of them, were carried out.

- 1) T_1 uses NE and T_2 uses PE.
- 2) T_1 uses PE and T_2 uses NE.
- 3) T_1 starts using NE and T_2 starts using PE, then change to PE or NE, respectively.
- 4) T_1 starts using PE and T_2 starts using NE, then change to NE or PE, respectively.
- 5) T_1 uses NE always and T_2 uses combination of PE-NE.
- 6) T_1 uses PE always and T_2 uses NE-PE.
- 7) T_1 uses combination of PE-NE and T_2 uses NE always.
- 8) T_1 uses combination of NE-PE and T_2 uses NE always.

Observe that method to strategies election is changed at inning 4 or 7. Observe that the change of selection of strategies occurs at the first middle inning 4th or at the first late inning 7th, and if needed at extra 9th inning. Considering the results in Tables 2.12 – 2.13 Figures 2.16 – 2.17 (item 1-2), the team that uses NE for selection of strategies in Baseball gaming, either for defense or offensive role, has advantage over the team that uses PE. In Tables 2.14 – 2.15 and Figures 2.18 – 2.19 (items 3 – 4), when a team, as soon as it is losing, changes strategy from NE to PE, or vice-versa, the results illustrate that the change is beneficial to the team because the score is closing, and sometimes the team that is losing can overcome the score. Tables 2.16 – 2.19 and Figures 2.20 – 2.23 show results (items 5 – 8) when one team fixes the strategy analysis and the other change and this last obtained an advance.

Table 2.12 Examples of games scores in item (1).

| # | Score | Inning | Winner |
|-----|-------|--------|----------------|
| 1 | 14 | 7 | T ₁ |
| 2 | 1 | 5 | T ₂ |
| 3 | 1 | 3 | T ₂ |
| 4 | 16 | 9 | T ₁ |
| 5 | 3 | 4 | T ₂ |
| ... | | | |
| 196 | 8 | 5 | T ₁ |
| 197 | 11 | 2 | T ₁ |
| 198 | 2 | 7 | T ₂ |
| 199 | 7 | 3 | T ₁ |
| 200 | 5 | 6 | T ₂ |

Table 2.13 Examples of games scores in item (2).

| # | Score | Inning | Winner |
|-----|-------|--------|----------------|
| 1 | 8 | 5 | T ₁ |
| 2 | 2 | 1 | T ₁ |
| 3 | 3 | 2 | T ₁ |
| 4 | 5 | 6 | T ₂ |
| 5 | 5 | 10 | T ₂ |
| ... | | | |
| 196 | 6 | 4 | T ₁ |
| 197 | 0 | 4 | T ₂ |
| 198 | 1 | 4 | T ₂ |
| 199 | 2 | 4 | T ₂ |
| 200 | 5 | 13 | T ₂ |

Table 2.14 Examples of games scores in item (3).

| # | Score | Inning | Winner |
|-----|-------|--------|----------------|
| 1 | 11 | 15 | T ₂ |
| 2 | 4 | 10 | T ₂ |
| 3 | 4 | 5 | T ₂ |
| 4 | 4 | 19 | T ₂ |
| 5 | 5 | 3 | T ₁ |
| ... | | | |
| 196 | 0 | 4 | T ₂ |
| 197 | 13 | 0 | T ₁ |
| 198 | 4 | 2 | T ₁ |
| 199 | 1 | 6 | T ₂ |
| 200 | 4 | 7 | T ₂ |

Table 2.15 Examples of games scores in item (4).

| # | Score | Inning | Winner |
|-----|-------|--------|----------------|
| 1 | 9 | 5 | T ₁ |
| 2 | 14 | 2 | T ₁ |
| 3 | 8 | 1 | T ₁ |
| 4 | 0 | 2 | T ₂ |
| 5 | 8 | 7 | T ₁ |
| ... | | | |
| 196 | 8 | 10 | T ₂ |
| 197 | 16 | 13 | T ₁ |
| 198 | 3 | 8 | T ₂ |
| 199 | 8 | 6 | T ₁ |
| 200 | 6 | 0 | T ₁ |

Table 2.16 Examples of games scores in item (5).

| # | Score | Inning | Winner |
|-----|-------|--------|----------------|
| 1 | 7 | 4 | T ₁ |
| 2 | 13 | 2 | T ₁ |
| 3 | 9 | 6 | T ₁ |
| 4 | 11 | 8 | T ₁ |
| 5 | 8 | 9 | T ₂ |
| ... | | | |

Table 2.17 Examples of games scores in item (6).

| # | Score | Inning | Winner |
|-----|-------|--------|----------------|
| 1 | 5 | 11 | T ₂ |
| 2 | 5 | 1 | T ₁ |
| 3 | 0 | 4 | T ₂ |
| 4 | 3 | 4 | T ₂ |
| 5 | 9 | 5 | T ₁ |
| ... | | | |

| | | | |
|-----|---|---|----------------|
| 196 | 6 | 2 | T ₁ |
| 197 | 0 | 1 | T ₂ |
| 198 | 1 | 4 | T ₂ |
| 199 | 2 | 6 | T ₂ |
| 200 | 5 | 7 | T ₂ |

Table 2.18 Examples of games scores in item (7).

| | | | |
|-----|---|----|----------------|
| 196 | 6 | 3 | T ₁ |
| 197 | 0 | 14 | T ₂ |
| 198 | 6 | 5 | T ₁ |
| 199 | 7 | 8 | T ₂ |
| 200 | 5 | 9 | T ₂ |

Table 2.19 Examples of games scores in item (8).

| # | Score | Inning | Winner |
|-----|-------|--------|----------------|
| 1 | 14 | 4 | T ₁ |
| 2 | 8 | 13 | T ₂ |
| 3 | 8 | 2 | T ₁ |
| 4 | 3 | 2 | T ₁ |
| 5 | 6 | 12 | T ₂ |
| ... | | | |
| 196 | 1 | 4 | T ₂ |
| 197 | 3 | 8 | T ₂ |
| 198 | 5 | 2 | T ₁ |
| 199 | 5 | 7 | T ₂ |
| 200 | 3 | 4 | T ₂ |

| # | Score | Inning | Winner |
|-----|-------|--------|----------------|
| 1 | 4 | 8 | T ₂ |
| 2 | 7 | 8 | T ₂ |
| 3 | 5 | 6 | T ₂ |
| 4 | 3 | 6 | T ₂ |
| 5 | 12 | 4 | T ₁ |
| ... | | | |
| 196 | 7 | 8 | T ₂ |
| 197 | 3 | 13 | T ₂ |
| 198 | 12 | 9 | T ₁ |
| 199 | 6 | 7 | T ₂ |
| 200 | 13 | 8 | T ₁ |

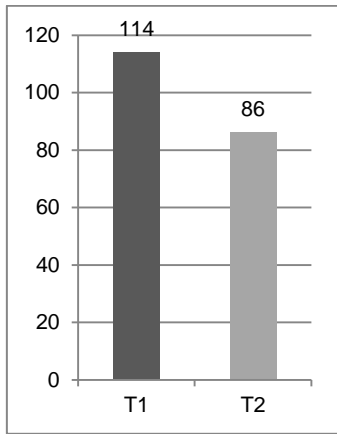


Figure 2.16 Percentage of winning of T₁ vs. T₂ in item (1).

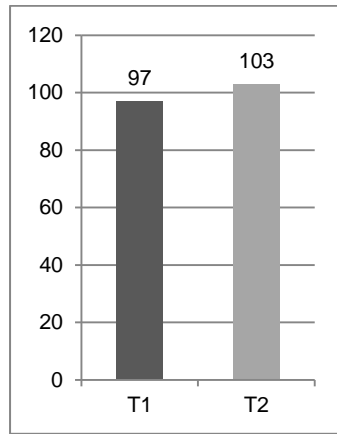


Figure 2.17 Percentage of winning of T₁ vs. T₂ in item (2).

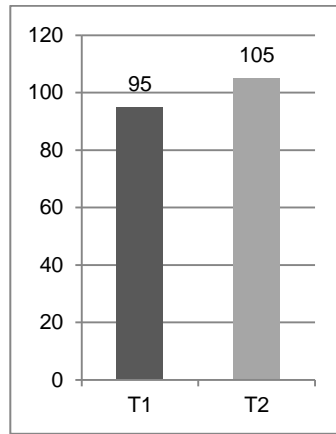


Figure 2.18 Percentage of winning of T₁ vs. T₂ in item (3).

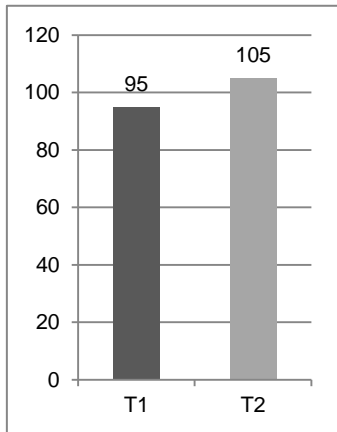


Figure 2.19 Percentage of winning of T₁ vs. T₂ in item (4).

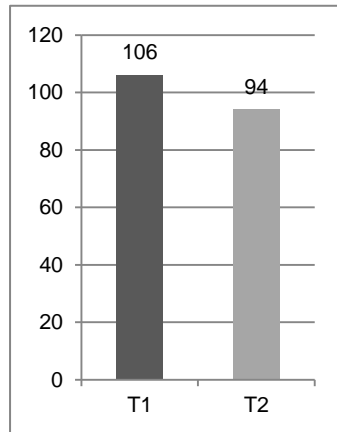


Figure 2.20 Percentage of winning of T₁ vs. T₂ in item (5).

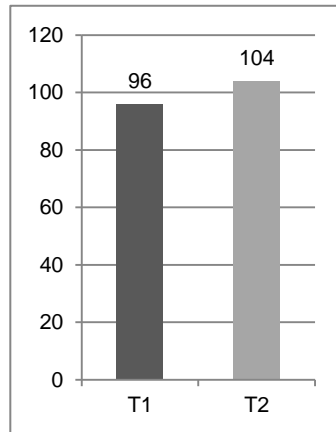


Figure 2.21 Percentage of winning of T₁ vs. T₂ in item (6).

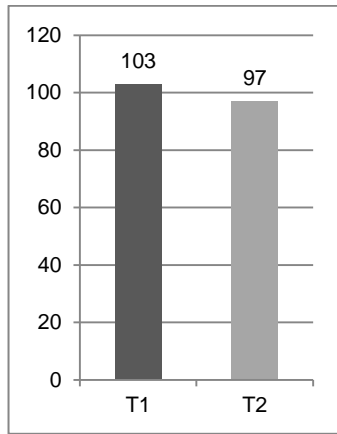


Figure 2.22 Percentage of winning of T₁ vs. T₂ in item (7).

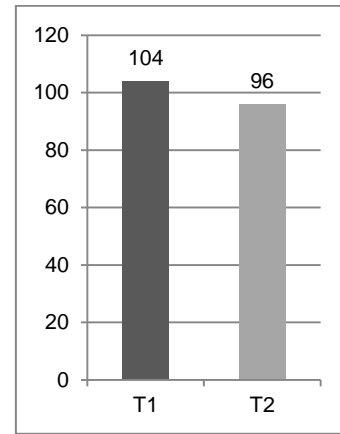


Figure 2.23 Percentage of winning of T₁ vs. T₂ in item (8).

Next, analysis focuses when T₁ and T₂ change their strategies selection method for items (3 - 8). In Figure 2.24, the result shows when T₁ begins NE and T₂ PE. Both change selection of strategies method NE-PE or PE-NE when losing. In Figure 2.24 (A) T₂ changes PE-NE in the 4th inning and the score has increased; in the 7th inning the T₁ changes NE-PE maintaining the score. Figure 2.24 (B) shows the percentage of increase, no increase and score closeness when teams change the selection of strategy. For T₁ the 20% increased, 35% did not increase and 45% closed the score when it changed NE-PE. Furthermore, for T₂ the 55% increased, 25% did not increase and 25% closed the score when it changed PE-NE. Observe that in some cases both teams change more than once.

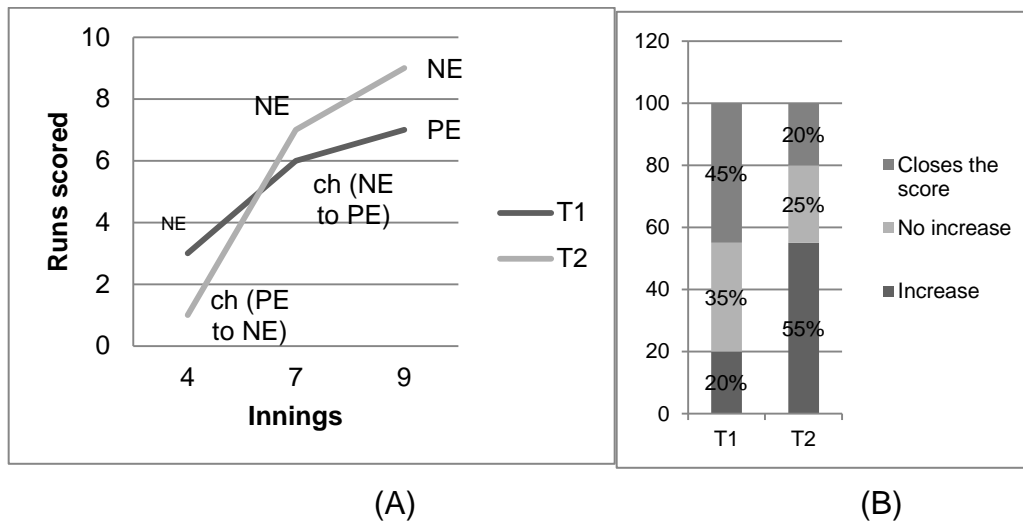


Figure 2.24 Analysis of change of strategy technique of teams (3).

Figure 2.25 illustrates when T_1 begins PE and T_2 NE, then both change strategy. In Figure 2.25 (A) T_1 changes PE-NE in the 4th inning and the score is improved; in the 7th inning T_1 changes NE-PE and the score is unimproved. Figure 2.25 (B) shows the percentage of increase, not increase and score closing when team change selection of strategies method: for T_1 60% increased, 15% did not increase and 25% score closing when there was change PE-NE; for T_2 , moreover, 30% increased, 30% did not increase and 40% closed the score when change NE-PE.

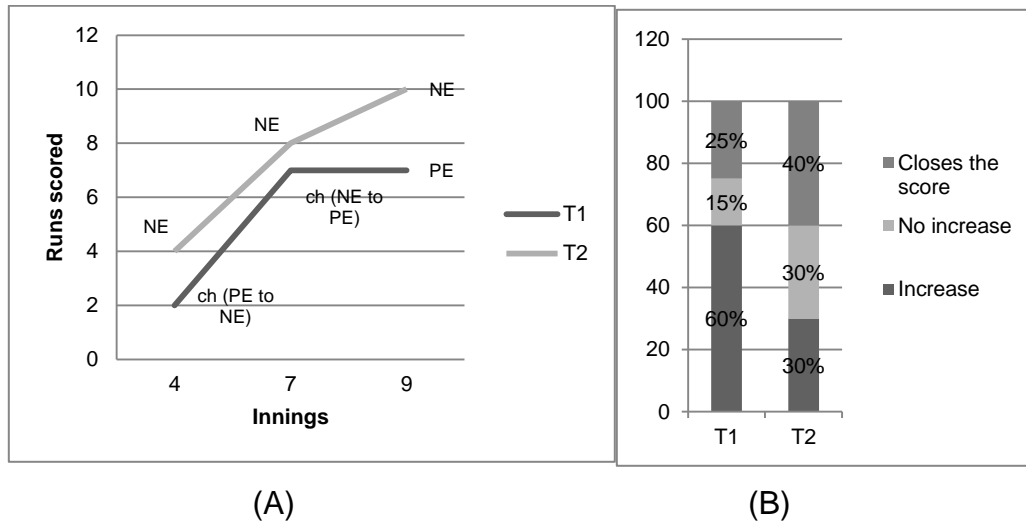


Figure 2.25 Analysis of change of strategy technique of teams (4).

In Figure 2.26 T₁ fixes NE and T₂ begins PE and can change strategy selection. In Figure 2.26 (A) T₂ changes PE-NE in the 4th inning improving score; in the 7th inning T₁ maintains NE and makes some score improvements. Figure 2.26 (B) shows the percentage of increase, not increase and score closing when team changes selection of strategy. For T₂ 60% increased, 15% did not increase and 25% score closing when it changes PE-NE.

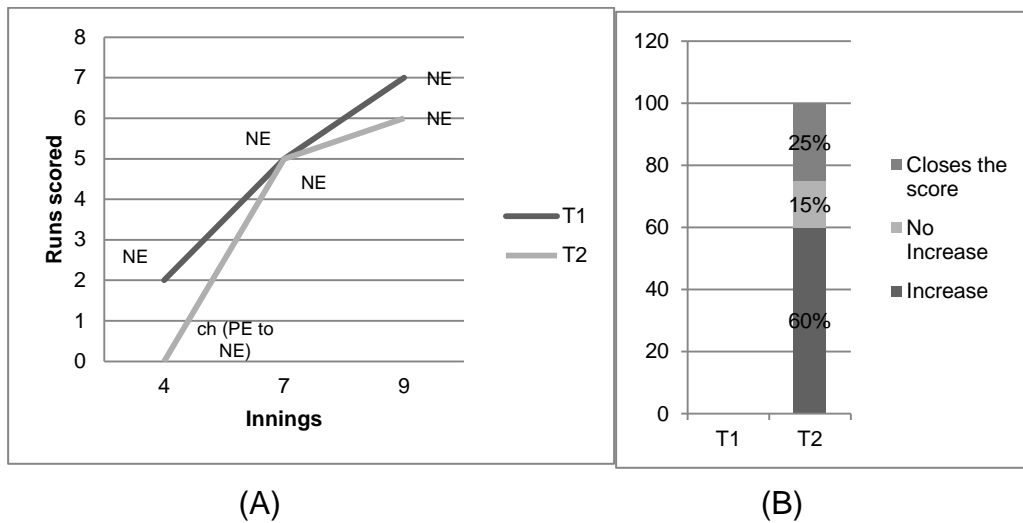


Figure 2.26 Analysis of change of strategy technique of teams (5).

In Figure 2.27 T₁ fixes to PE and T₂ begins NE and can change selection of strategies. In Figure 2.27 (A) T₂ changes NE-PE in the 7th inning and it did not improve the score. Figure 2.27 (B) shows the percentage of increase, did not increase and score closing when team changes selection of strategies method: for T₂ 35% increased, 35% did not increase and 35% score closing by change.

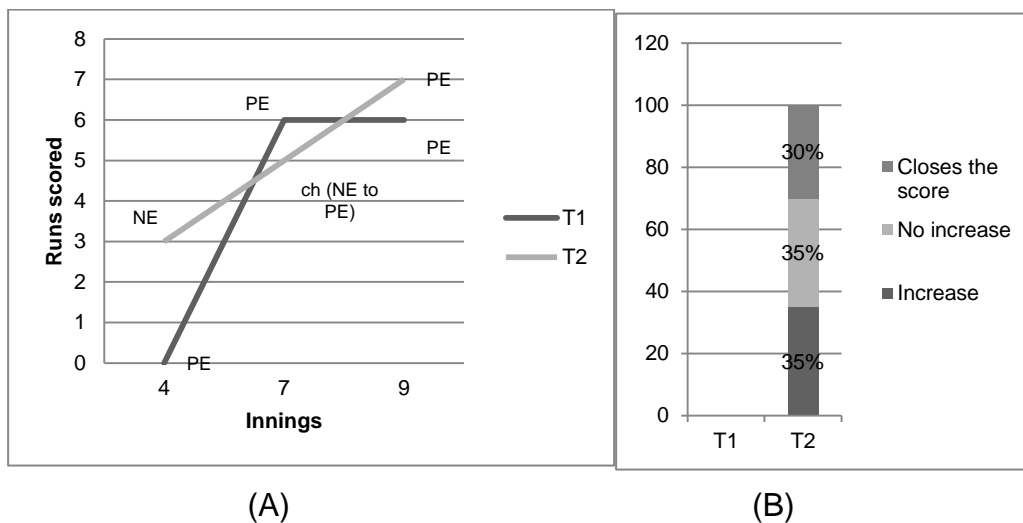


Figure 2.27 Analysis of change of strategy technique of teams (6).

In Figure 2.28 T_2 fixes NE and T_1 begins PE and can change selection of strategies. In Figure 2.28 (A) T_1 changes PE-NE in the 4th inning and improves the score. Figure 2.28 (B) shows the percentage of increase, not increase and score closing by team change; for T_1 60% increased, 15% did not increase and 25% score closing when selection of strategies PE-NE was changed.

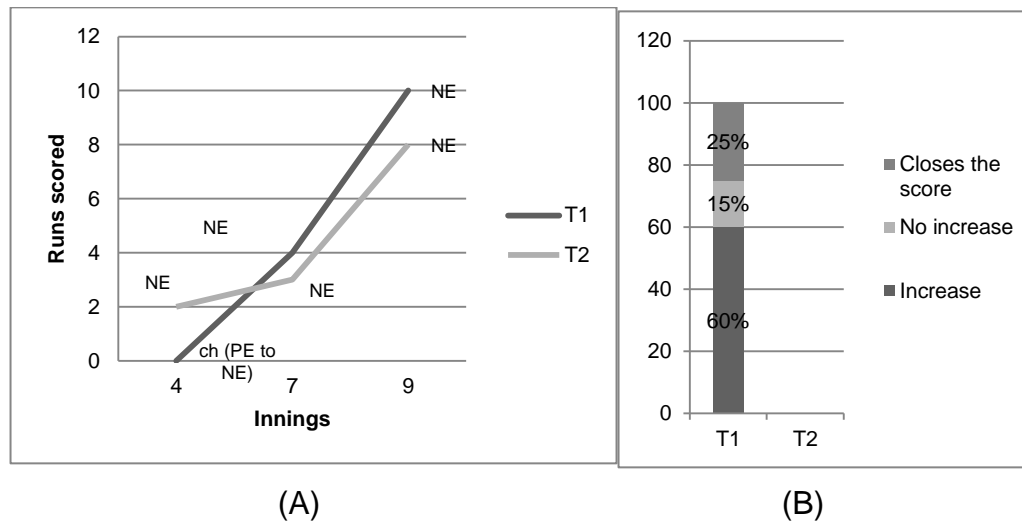
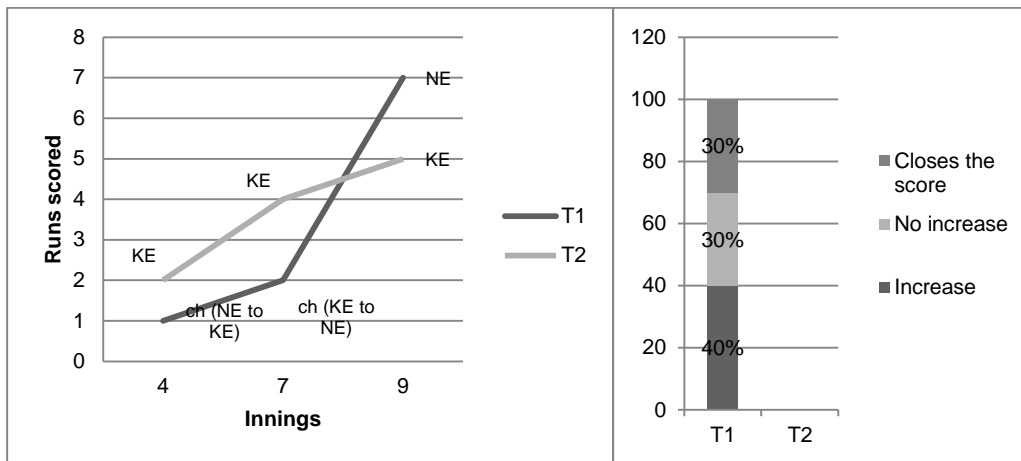


Figure 2.28 Analysis of change of strategy technique of teams (7).

In Figure 2.29 T_2 fixes PE and T_1 begins NE and can change selection of strategies method. In Figure 2.29 (A) T_1 changes NE-PE in the 4th inning, and does not improve the score, T_1 changes NE-PE in the 7th inning improving his score. Figure 2.29 (B) shows percentage of increase, did not increase and score closing when team changes selection of strategies method; for T_1 40% increased, 30% did not increase and 30% score closing when selection of strategies method was changed.



(A)

(B)

Figure 2.29 Analysis of change of strategy technique of teams (8).

The results obtained revealed the positive impact, the percentage of gain or loss, the change of strategy selection for a team regarding items (3 – 8); thus the advantage of using NE or PE for strategy selection in a Baseball match.

2.7.3 Nash Offensive versus Pareto Efficiency Defensive

Now, we analyze the impact of NE for offensive role and PE for defensive role, to observe whether NE/PE is well behaved for the specific defensive/offensive role, versus the usage of NE or PE during the whole match without any change. Two hundred simulations of Baseball matches per each following item were performed (see results in Figures 2.30 – 2.32).

- (a). Both T_1 and T_2 use NE for offensive role and PE for defensive role.
- (b). T_1 uses NE for offensive role and PE for defensive role and T_2 only uses NE.
- (c). T_1 uses NE for offensive role and PE for defensive role and T_2 only uses PE.

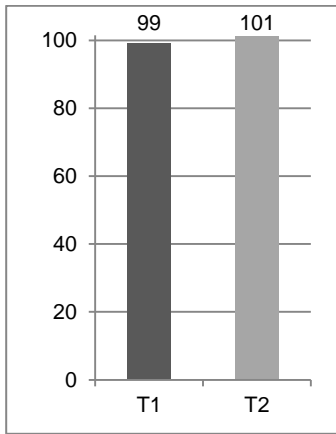


Figure 2.30 Percentage of winning of T₁ vs. T₂ in item (a).

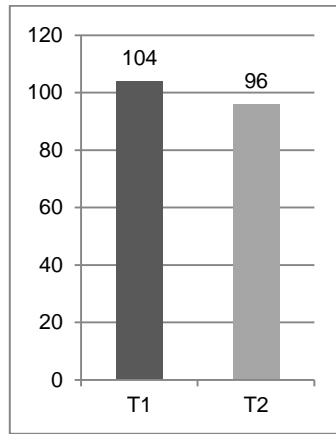


Figure 2.31 Percentage of winning of T₁ vs. T₂ in item (b).

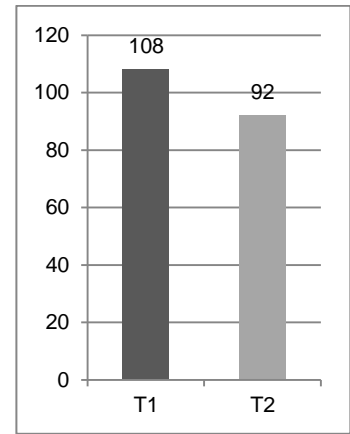


Figure 2.32 Percentage of winning of T₁ vs. T₂ in item (c).

From the results illustrated in in Figures 2.30 – 2.32, it may be concluded that the teams using NE for the offensive role and PE for defensive achieve better performance than those that only use one method for selection of strategies gaming any of the roles.

The percentage of strategy profiles being likewise PE and NE is 58%. The remaining 42% is of different strategy profiles (see Figure 2.33). In addition, when the analysis determines that NE strategic profiles should be done for gaming, the percentage really practiced is 71% (see Figure 2.34), whereas for PE, 46% is practiced (see Figure 2.35).

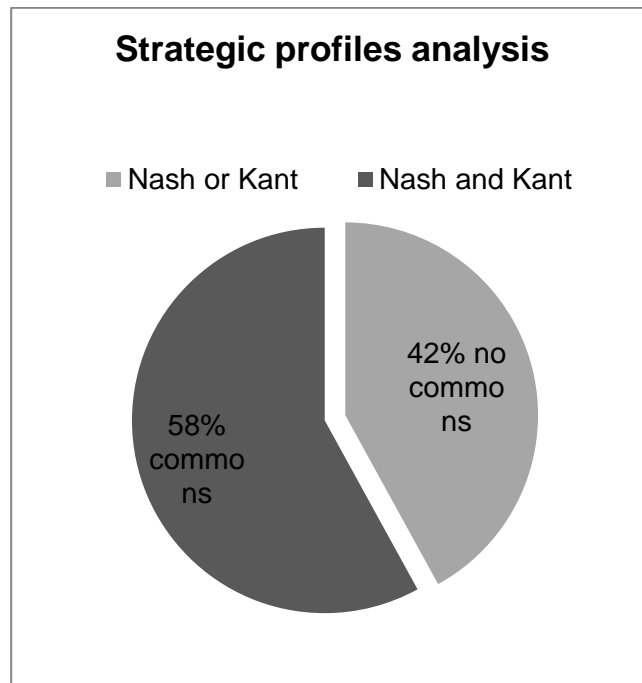


Figure 2.33 Common strategic profiles between NE and PE.

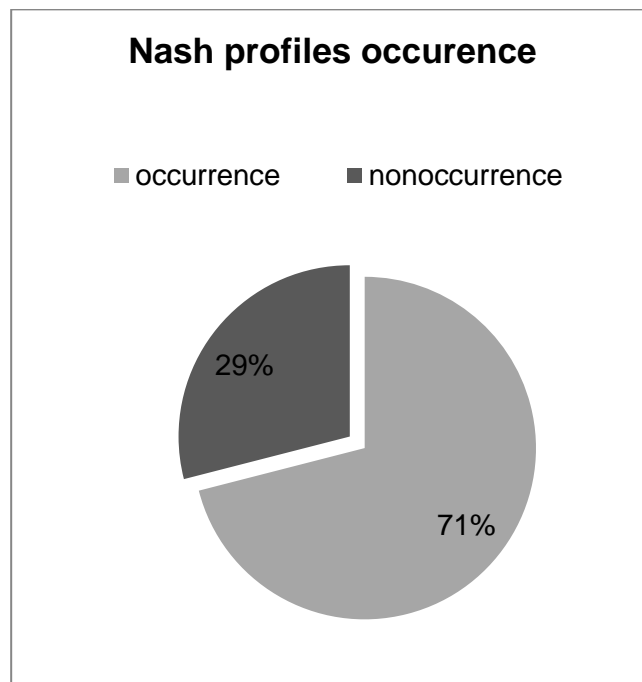


Figure 2.34 Percentage of occurrence of NE profiles.

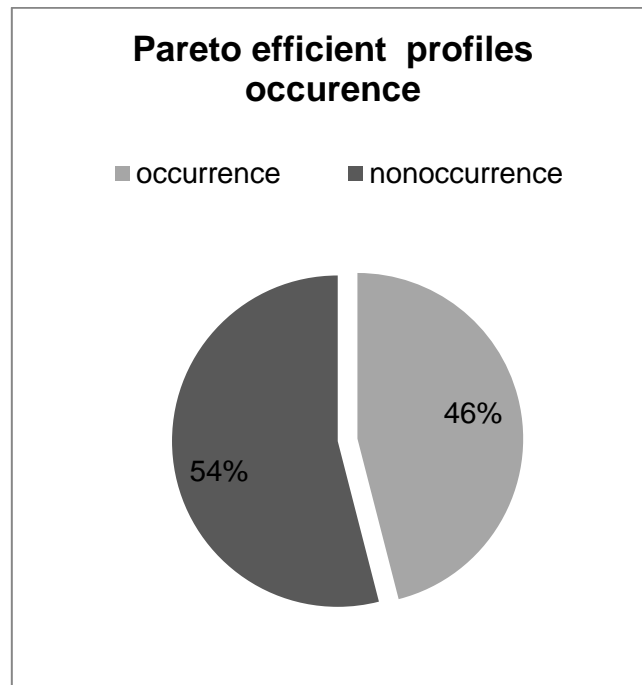


Figure 2.35 Percentage of occurrence of Pareto efficient profiles.

As conclusion of this analysis, beyond to consider if strategic profiles are Pareto efficient or NE, there are factors which influence in the Baseball gaming, e.g., the SO of Baseball plays. The SO of plays determines the nature of the game, i.e., there are Baseball plays that are more likely to occur than others, hence, if a Pareto efficient strategic profile is harder to occur than a NE strategic profile, so, a NE profile will occur more often than a Pareto efficient profile. This is because Pareto efficient profiles are those where the players perform the best plays but these plays are more difficult to occur. The major examples are the sacrifices plays. For players who are sluggers (home run hitters), they would prefer a home run rather than a sacrifice fly but, a sacrifice fly is more likely to happen than home run even if players are sluggers.

Aggressive or conservative playing depends if the game is on initial, middle, or late innings. The first innings main goal is going-ahead by means of an aggressive playing without wasting outs by sacrifice bunts. Middle innings often determines the game character, in front of an aggressive style to play conservative is recommended. In the late innings and up on the scoreboard an aggressive play is recommended, but down on the

scoreboard, a conservative play is better option. Up on the scoreboard, play more recklessly to increase the difference on the scoreboard is recommended, but down on the scoreboard, a conservative style works, to keep the number of outs and the players on base. Without outs, a small difference on the scoreboard and in the late innings, a conservative play should be applied for scoring few more runs and having one or two outs play aggressive to reach at least one run more [3, 68].

2.8 Hungarian Selection: Team Performance Comparison

In MLB players selected using HA are listed in Table 2.20 and the probability matrix obtained for offensive actions is presented in Table 2.21.

Table 2.20 Player selected using Hungarian Algorithm.

| Player | Original position | Team | Assigned position | Benefice |
|-------------------|-------------------|---------|-------------------|---------------|
| Mark Teixeira | 1B | Yankees | C | 1.1187 |
| Adrián González | 1B | Red Sox | 1B | 0.9915 |
| Dustin Pedroia | 2B | Red Sox | 2B | 1.0352 |
| Robinson Cano | 2B | Yankees | SS | 0.8308 |
| Derek Jeter | SS | Yankees | 3B | 0.7256 |
| Cody Ross | RF | Red Sox | LF | 0.7446 |
| David Ortiz | DH | Red Sox | CF | 1.0607 |
| Curtis Granderson | CF | Yankees | RF | 0.9595 |
| Nick Swisher | RF | Yankees | DH | 0.9023 |
| CC Sabathia | SP | Yankees | SP | 1.1444 |
| TOTAL | | | | 9.5134 |

Table 2.21 The probability obtained for each player.

| | H | HR | 2B | SF | SH | SB | GDP | SO | BB |
|---|---------------|---------------|---------------|---------------|---------------|---------------|--------|---------------|---------------|
| 1 | 0.2996 | 0.0310 | 0.0764 | 0.0145 | 0.0000 | 0.0000 | 0.0186 | 0.1674 | 0.0641 |
| 2 | 0.2895 | 0.0266 | 0.0693 | 0.0107 | 0.0018 | 0.0355 | 0.0160 | 0.1066 | 0.0853 |
| 3 | 0.2668 | 0.0462 | 0.0714 | 0.0126 | 0.0021 | 0.0042 | 0.0231 | 0.2710 | 0.0882 |
| 4 | 0.3179 | 0.0710 | 0.0802 | 0.0093 | 0.0000 | 0.0000 | 0.0185 | 0.1574 | 0.1728 |
| 5 | 0.2156 | 0.0458 | 0.0515 | 0.0229 | 0.0000 | 0.0038 | 0.0210 | 0.1584 | 0.1031 |
| 6 | 0.2812 | 0.0473 | 0.0689 | 0.0029 | 0.0000 | 0.0043 | 0.0316 | 0.1377 | 0.0875 |
| 7 | 0.2919 | 0.0203 | 0.0432 | 0.0014 | 0.0081 | 0.0122 | 0.0324 | 0.1216 | 0.0608 |

| | | | | | | | | | |
|------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|
| 8 | 0.2018 | 0.0629 | 0.0263 | 0.0102 | 0.0015 | 0.0146 | 0.0073 | 0.2851 | 0.1096 |
| 9 | 0.2340 | 0.0385 | 0.0577 | 0.0080 | 0.0016 | 0.0032 | 0.0144 | 0.2260 | 0.1234 |
| Avg | 0.2665 | 0.0433 | 0.0606 | 0.0103 | 0.0017 | 0.0086 | 0.0203 | 0.1812 | 0.0994 |

A comparative analysis mixing different techniques applied to Baseball teams was performed. A total of six hundred computer simulations of Baseball matches were performed for the next items, **one hundred (100) simulations** each.

1. Team 1 (T_1) uses HA and NE; versus Team 2 (T_2) uses NE.
2. T_1 uses HA and NE; versus T_2 uses PE.
3. T_1 uses HA and PE; versus T_2 uses NE.
4. T_1 uses HA and PE; versus T_2 uses PE.
5. T_1 uses HA and NE versus T_2 uses HA and PE.
6. T_1 uses HA and PE versus T_2 does not use any method.

The results in Figure 2.36 correspond when T_1 uses HA to assign player-positions and NE for strategic analysis while T_2 uses only NE; T_1 achieved more victories 55/45 and in a total of 5 Baseball matches reached *extra innings*. Figure 2.37 shows the results when T_1 uses HA selection and NE while T_2 only uses PE; T_1 won more victories 60/40 and 5 Baseball matches reached *extra innings*. Figure 2.38 shows the results when T_1 uses HA selection and PE while T_2 only uses NE; T_1 won more victories 53/47 and 4 Baseball matches reached *extra innings*. Figure 2.39 shows the results when T_1 uses HA selection and PE while T_2 only uses PE; T_1 scored more victories 59/41 and 8 Baseball matches reached *extra innings*. Figure 2.40 shows the results when T_1 uses HA selection and PE while T_2 uses PE; T_1 won 54/46 and 4 Baseball matches extended to *extra innings*. Figure 2.41 shows the results when T_1 uses HA selection and PE while T_2 uses PE; T_1 won 61/39 victories and only 6 Baseball matches reached *extra innings*.

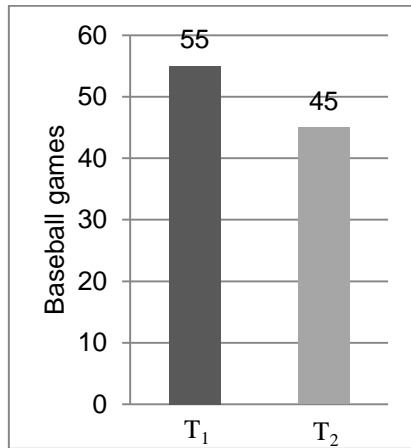


Figure 2.36 HA-NE *versus* NE.

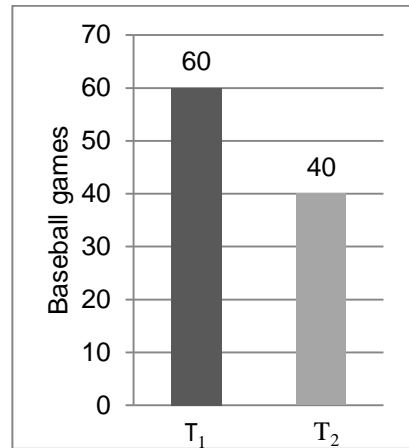


Figure 2.37 HA-NE *versus* PE.

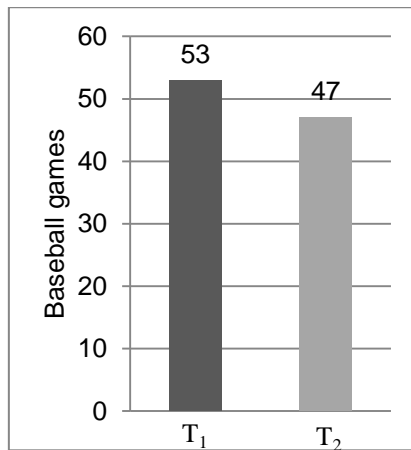


Figure 2.38 HA-PE *versus* NE.

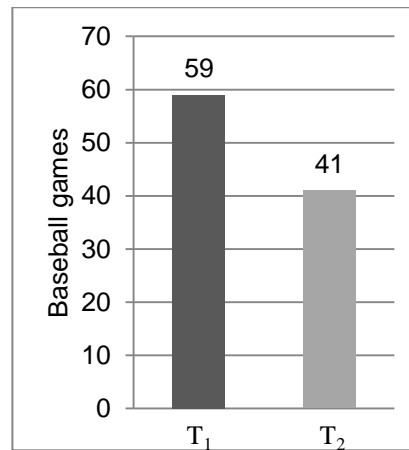


Figure 2.39 HA-PE *versus* PE.

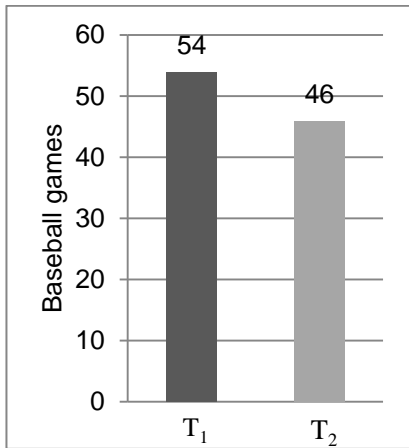


Figure 2.40 HA-NE versus HA-PE.

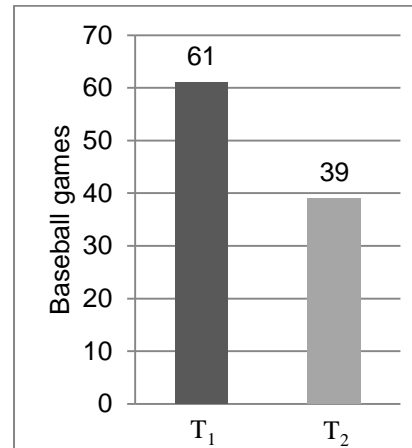


Figure 2.41 HA versus No-Method.

Next follows a comparative analysis among Baseball teams that use HA to assign player-positions and NE or PE to select strategies.

Figure 2.42 illustrates the behavior when T₁ only uses HA with NE while T₂ uses PE, NE and HA with PE. The T₁ performance is superior to all other techniques used by T₂ and the T₂ performance improves by changing to a better technique.

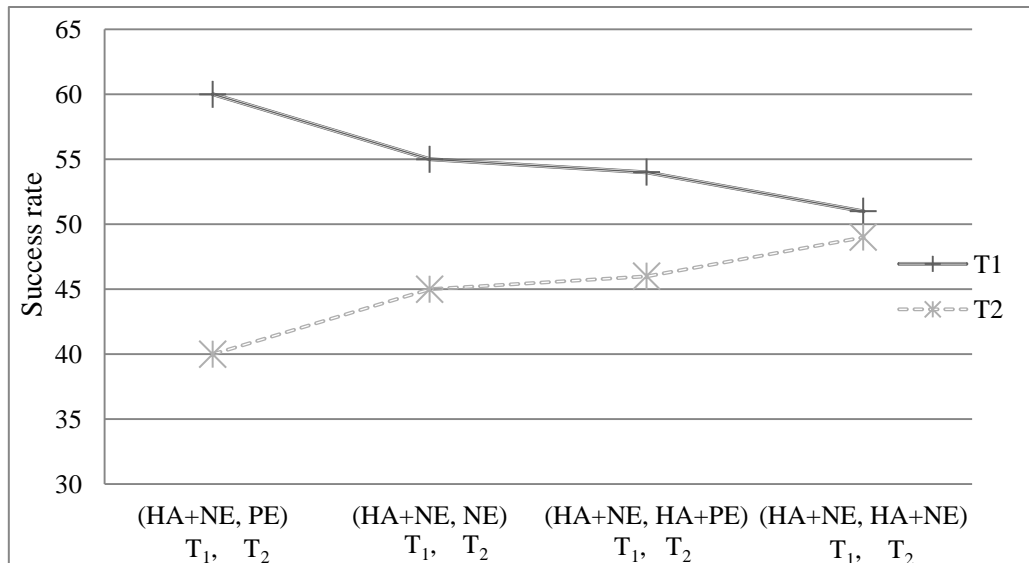


Figure 2.42 T₁ only uses (HA+NE) and T₂ uses (PE, NE, HA+PE, HA+NE).

Figure 2.43 describes the use of different techniques by T_2 while T_1 only uses HA+PE. The T_1 performance decays while the T_2 performance increases and even exceeds the T_1 performance when T_2 uses HA+NE.

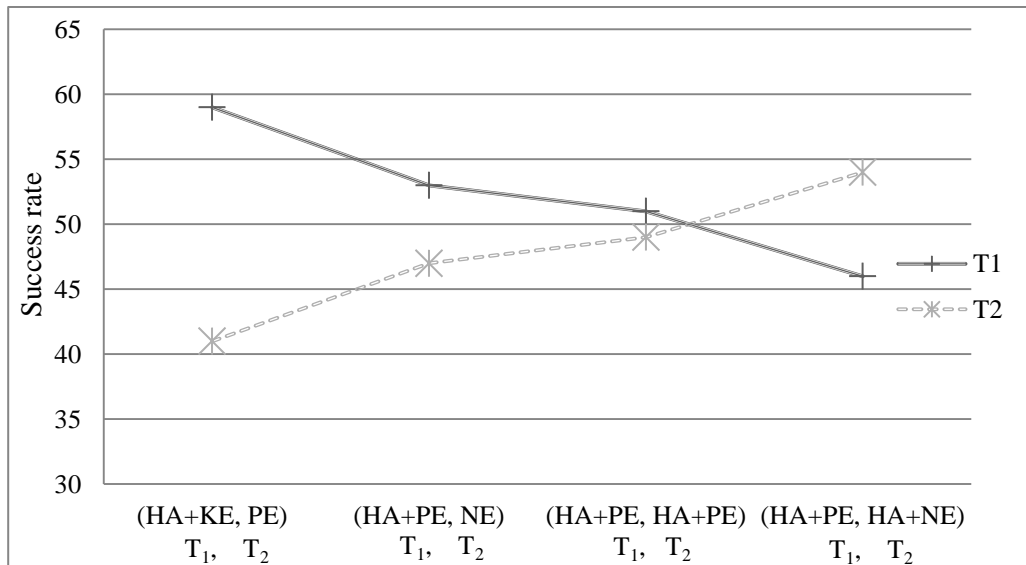


Figure 2.43 T_1 only uses (HA+PE) and T_2 uses (PE, NE, HA+PE, HA+NE).

As In summary, the team selected by HA and using NE for selection of strategies showed a better performance than any other team which used any other combination of methods. In addition, a comparison of when T_1 used HA for player-positions assignment when T_2 did not use HA or perform any selection of strategies shows that the T_1 performance is better.

2.9 Discussion

On decision making being supported by computer tools, for single or multi-player games, GT formal approach [82], having as major purpose the economics modeling on markets, trade and financial issues, nowadays applied to a growing number of areas such as politics, biology, sociology, information technologies and engineering.

In the examples presented about the analysis of strategies, different circumstances of a Baseball match are analyzed using Nash equilibrium and Pareto efficiency for making strategic choices. Particularly, one strategy profile of sacrifice plays was finding by using Nash equilibrium. Qualitative analysis [68] and statistical studies [106] about the pertinence of sacrifice plays in a Baseball match explain the best moment to apply them. The equilibrium analysis on the strategy profiles of sacrifice plays in a Baseball match being supported by computer simulations [3], found that these profiles fit the Nash profiles when circumstances of Baseball match are, the last innings, the match score tied, one player on third base and one/none out(s) in the inning. For these circumstances sacrifice plays are applied opportunely to reach the best result. The convergence to these profiles is by means of increase the probability of occurrence of these plays; in practice, the manager should indicate his players try to perform these plays, so the probability to these plays is increased. Moreover, according to our experimental results, the probability of convergence to the strategy profiles of sacrifice plays is over 60 percent, at the last innings and tied score. We claim a probabilistic convergence to desired profiles because the stochastic nature of the Baseball game, many uncertain factors –beyond the team’s control.

As we discuss previously, PE allows the theoretical Pareto-efficient, so optimal design of strategies. However, in real human Baseball matches, the theoretical design cannot occur by the presence of uncertain factors –beyond the team’s control. Whereby, the use of theoretical Pareto efficient strategies under some circumstances in Baseball game is low feasible. In a Baseball match there are many uncertain factors such as, human ways of pitching, running and batting, or natural factors like wind speed or the height of the place, which affect the playing performance. The stochastic nature of the Baseball game is well-modeled by our approach, and the convergence to some identified equilibrium points observes the statistics from real matches.

A convergence method proposed by Clempner and Poznyak [24], finds an equilibrium strategy profile using a vector Lyapunov-like function in strictly dominated games, where strategy profiles with dominated strategies are deleted. A Lyapunov strategy profile (point) is a Nash equilibrium point. The convergence method is applied to the prisoner’s

dilemma and battle of the sexes, two players and two strategies games. For the future, it would be interesting analyze the Lyapunov-like function for convergence to Lyapunov strategy profiles on multi-players games, like Baseball that have many more than two players and strategies.

2.9.1 Nash Equilibrium Computing Complexity

In multi-player symmetric game where each player has a small number of strategies, a polynomial-time algorithm for finding NE is presented [87]. Even if the strategies are NE profiles and efficiently found, usual implementation is hard to do [69, 87]. The proposed algorithm for Baseball match analysis makes deviations of profiles to rule out the profiles that are dominated for all the NE profiles and are find out by using NE payoff matrix. This is low cost with computational complexity k^n , with k is the number of strategy profiles and n is the number of players. In the worst case, k is around 512 strategy profiles and n is 9, therefore the computer complexity is polynomial time.

In addition NE approximation in an additive and multiplicative sense, with n pure strategies per player in games with $r \geq$ players is in [55]. NE in discrete routing games is investigated in [45], in a discrete routing game, each of n selfish users employs a *mixed strategy* to ship her *traffic* over m parallel *links*.

2.9.2 Cooperation Management

For the sake of collective convenience among the bidding agents, a kind of equilibrium is desirable into such complex process. A mixed-strategy by reinforcement learning allows changing the price strategy in a game over time to two sellers so a theoretical NE convergence is obtained [10]. The interaction among employees of a provider firm and the ones of an outsourcing firm, whose should share knowledge and skills to work as a team but that might be antagonistic each other, is in such a way that whenever the degree of complimentary knowledge among the employees is high, a better payoff is

achieved up to the provider and outsourcing firm's top management enforces cooperation than when they decide not to do [9].

Cooperation in players' strategic interactions in finite games with independent actions and equal distributed random payoffs over continuous functions is such that per realization, players show all the payoffs values and simultaneously choose strategies [27]. The NE and Pareto optimality results suggest that cooperation becomes increasingly advantageous in increasingly complex situations when actors have increasingly numerous of possible responses to the strategic actions of others.

In the so called tragedy of the commons problem a set of fishers must expend labor on a lake to catch fish. Each fisher has a utility function over fish caught and labor expended [95]. Kantian equilibrium (KE) formalism supports the design of Pareto models in economy and, the theoretical optimal, team collaboration [94]. KE guarantees that the each other commit allows the theoretical optimum on team collaboration [94, 95].

In Kantian equilibrium (KE) [95] all players have a common strategy space S , so the normal form game is $G = (S; u_1, \dots, u_n)$; a strategy profile (s_1, \dots, s_n) is Kantian if equation (2.20) holds:

$$u_i(s_1, \dots, s_n) \geq u_i(\alpha(s_1, \dots, s_n)) \quad \forall i \in P, \alpha \in \mathbb{R}_+ \quad (2.20)$$

All of the player's action value is weighted by the same factor α . This is community cooperation in theoretically equal conditions and no one player takes advantage from any other. By KE usage every player is applying the Pareto efficient best own strategy from a cooperative perspective, and there is at least one strategy profile for a game in normal form that fits Kantian equilibrium, as for NE. For KE, all players get the maximum profit, in fact the player changes his strategy if and only if each player changes its strategy by the same multiplicative factor α , we interpret α as a change in the strategy profile for all players, and perhaps, we do not fit strictly with real definition on (2.20) but we claim at least that the profiles be Pareto efficient. Nevertheless, due to the lack of interpretation of KE in this kind of games, we can interpretation the use of KE as previously.

Payoff matrix formalism. To compute NE for finite strategic game the covariance matrix adaptation evolution strategies (CMA-ES), the particle swarm optimization (PSO), and the differential evolution (DE) were applied [89]. An algorithm to obtain the NE of n -player matrix games with stochastic reinforcement learning has the potential to solve within large player-action spaces [80]. The solution approach is related to matrix games with discounted and average reward stochastic games.

2.9.3 Couple and Team Formation

In GT, a problem of finding a stable matching between two sets of elements is known as the stable marriage problem (SMP) [46]. In this problem, we have a set of n men and n women where each person has his/her own preference list of the persons that he/she wants to marry. The goal is to have a set of stables marriages, such that, there are no two persons of opposite set who would prefer other person than his/her current partner. Gale and Shapley in [46] proved that there is a stable set of marriages. In the case of Baseball, the solution of SMP does not satisfy the player-positions assignment because SMP emphasizes finding solutions by couple rather than by group, i.e., the couple (*player, Baseball-position*) is attended individually without regard the others couples. On the opposite, HM allows to find out the best couples (*player, Baseball-position*) thinking of forming the best team. The best Baseball-positions couples are not the assigned sole regarding the best statistics-player, but such that all Baseball-positions-player are attended in a way that emerging team guarantees the best playing performance.

This thesis applied HA for Baseball players' selection and NE or PE to coordinate the Baseball team. From simulations of Baseball matches, we showed that the best team performance is improved by the use of HA and NE, thus, we claim that not only with a suitable players' selection, is enough, it requires a strategic analysis that guides the team to the victory. On the other hand, the correct way to make players' selection has significant importance in the team performance.

Nash equilibrium and Pareto efficiency for selection of strategies in Baseball gaming, jointly used with the Hungarian method to choose a Baseball team, fit the aim of improv-

ing the team performance. Actually, according to our set of computer simulation tests the combination HM+NE methods produces the best Baseball team performance during match playing. HE+PE, theoretical best, frequently cannot be practiced by empirical reasons ever present in real matches, e.g., the home run is one of the best Baseball offensive options and under some circumstances, it belongs to Pareto efficient profiles. If it happened is the best option for the team, but has a low statistical occurrence in practice. Whenever the risk of trying to perform a home run is less than the benefit that the team could get then, it is a good option to try to perform it; otherwise it is better to use another option. We emphasize, on these kinds of games not always the best theoretical options can be practiced.

On the task assignment approaches that uses HA such as [42, 51] may not require a strategic analysis after the assignment, but in Baseball, the assignment is one problem to attend, but, there exist factors during a match that requires a strong strategic analysis to find suitable strategies to achieve team victory.

2.9.4 Critical Zone Behavior in Biological and Physical Systems

Since the end of last century a growing interest is present in the analysis of complex networks having heterogeneous elements and positive and negative interactions, either directed or not, from diverse kinds. Fundamental in this emphasize is the need for understanding the structure and dynamics in the social networks as well as the strength capacity of current computers for simulating events in a real manner.

In the studies on biological, genomics, protein and metabolism networks was founded that all of these networks operate in critical zones, in between the order and disorder, where the homeostatic regulation is more effective, where little changes in the origins induces big changes on the effects.

Inside the study on biological dynamics is omnipresent the analysis on the series of elements like the nucleic bases in genetics sequences, amino acids in proteins, join to the electric cardiogram, electric encephalogram, and other records in the central nervous

system. In this last one has been found that like in other biological systems the central nervous system operates in critical zone, between the order and disorder.

In the area of collective games like Baseball, there are series of context free plays as well as independent from each other players' behavior. Up to the match score the manager do changes of gaming strategies. Thus, the systemic study on strategies changes is relevant. A first decision concerns the assignment of positions to the players based on each one capacity that has been studied by applying the so called Hungarian algorithm [19]. Other criteria are the permanence of change that the manager decides to use when the pitcher of any player's performance comes down. Beyond the manager's experience and intuition to change the player at any match moment, in this work we consider the criteria to alternate the strategies election by following PE or NE. Actually, a relevant question is what's on the quantitative criteria to define the need of change the gaming strategy? Results in previous sections give an advising about it.

2.10 Conclusion

In the top strategic game of Baseball, team's cooperation is essential for success. The usage of combined NE-PE strategy profile leads a Baseball team to choice strategies for building the match victory in such a way that every player's decision is by including from the other's ones. Formal account is normal form game and payoff matrix done, the algorithmic fundament is a context-free grammar and the match plays run random enough, so the simulator implementation is attaining similar scores to the human teams' ones in real life matches. The combined NE-PE, as the applied strategy from the team is losing, pushes to close or overcome the match score to the current advantaging team; this gaming style mixing Nash individual-self-centered strategy with a Pareto optimal strength the team gaming, and looks like to behavior in biological and physics networks as the correction to strength the system's behave.

Moreover, the results from computer simulations of Baseball matches show that, although the usual *non-cooperative* qualification to NE, it is a relative adjective, up to the real circumstance. In the context of a Baseball match, with several parameters out of the

players' and manager's control, NE allows identify strategy profiles for effective cooperation in real circumstances of Baseball gaming. The use of NE prevents to try plays or strategies with low statistical occurrence, so to avoid the risk to lose score points. It means that NE strategy profiles frequently include plays and strategies with higher statistical occurrence, so they are more feasible in real circumstances of matches. Furthermore, NE, by avoiding the risk to lose score points induces an effective cooperation for a team. On the other hand, PE formal account, it induces to choose the theoretically optimum strategy profiles. We observe that the best plays and strategies have low statistical occurrence, so few time to be practiced in real Baseball gaming circumstances. The Pareto efficient strategy profiles are less likely to occur than the Nash ones. Strategies in Pareto efficient profiles may be the most profitable but their probability of occurrence is low, and it moderates the use of Pareto efficiency to identify circumstances of cooperation in real circumstances of Baseball gaming.

The assignment problem is a complex task even in sport like Baseball. In this Chapter, we also used the HA for assigning a set of players to a set of Baseball positions in order to improve the team performance and for coordination NE and PE as strategic analysis. Results obtained using the Baseball simulator showed better performance to those teams that use HA and NE than those that use other mixing or lonely techniques. Therefore, it is important to have methods to make efficient the players' selection and methods to guide the team victory.

3

Simulation of American Football Gaming

3.1 Overview

Recently, the formal modeling and strategic analysis for support the matches gaming of multi-player sports like American Football (AF) or Baseball is growing [8, 52, 104]. These multi-player games have led investigations in areas of sport science [2, 61, 70], computer science, game theory [3], operation research [8, 104], simulation models [37, 52], among others.

In [104] an analysis of the forecasts of the outcomes of National Football League (NFL) games made by 31 statistical models with those of 70 experts was studied. A comparative of accuracy results that the experts and statistical systems in predicting game winners was not statistically significant. Similar approach in [8] presented a model for forecasting end-of-match exact scores in NFL games. The model uses a set of covariates based on past game statistics, to make predictions of the game results and exact scores.

In [52] a program capable to learn to play a game of imperfect information by observing its opponent's play is presented. The program uses an historical database of records of opponent's moves which effectiveness was tested playing against experts in AF, and similar learning curve as human opponent was reported.

In [37] presented a simulation-model for AF plays in which the representation of individual Football players' positions and velocities as a function of time, by means of Hill's equation of human motion and non-zero initial velocities. The Monte Carlo model is used to simulate the times history of the players' positions and velocities. In [59] studied a particular game situation in a college Football uses as decision criteria the maximum expected utility based on a *von Neuman-Morgentern* utility function, and as alternative option the stochastic dominance is applied.

The selection of strategies is an essential aspect to be considered for a whole AF automation. In this Chapter, we present a formulation using context-free-grammar for the algorithmic setting of AF; hence the corresponding finite state machine is described as well, doing emphasis in the strategic study of the behavior of the players in a game match.

3.1.1 Description

American Football is one of the top strategic games, played by two teams on a rectangular shaped field, 120 yards long by 53.3 yards wide, with goalposts in the end of the field. Each team has 11 players and a match lasts 1 hour divided in four quarters. The offensive team goal is advance an oval ball, by running or passing toward the adversary's end field [7, 22, 50]. The ways to obtain points are by advancing the ball, ten yards at least, until reach to the end zone for touchdown scoring, or kicking the ball such that it passes in the middle of the adversary's goalposts for a field goal, or by the defensive tackling the ball carrier in the offensive end zone for a safety. The offensive team should advance the ball at least ten yards in at most four downs (opportunities) to get four additional downs; otherwise the defensive team that is avoiding the ten yards advance, changes to the offensive role. The current offensive team advance starts from the last ball stop position. If the defensive catches the ball before a down is completed, it starts the offensive role at this position. A down ends by the most common circumstances that follow: when a pass is not successful, or when a player is tackled inside the field, or when a ball gets off the field.

The offensive team members follow: the quarterback is the offensive leader; half-backs/tailbacks do carrying the ball on running plays; centers, for snapping the ball to the quarterback; guards, for protect the quarterback from the tacklers; wider receivers' main goal is to catch passes thrown by the quarterback; tight ends, for doing function like the guards and wide receivers. In the defensive team, the linebackers are the defensive leaders; the principal role of the defensive team is to stop the running plays on the inside and outside; the defensive tacklers, for stopping the running plays; cornerbacks, line up outside the defensive formation and cover wide receivers; safeties, for stopping deep passes and running plays.

In multi-player football game, the team strategic reasoning is presented. The AF team members are encouraged to do the best individual actions, but they must cooperate for the best team benefit. The strategies are indicated by the team manager regarding on each player's profile as well as the specific match circumstances to obtain the most benefit [72]. A planned-strategy should include both the individual and the team motivation.

3.1.2 Strategy Thinking: Cooperation and Non-Cooperation

Strategies are organized and weighted actions practiced to obtain the maximum available profit up to the minimum effort [12, 41, 73]. Regarding the game rules, a player should determine the order and preference of his own actions and strategies joint to the threat embodied in the other players' strategies mind to obtain match success [31, 85, 91]. A successful result in matches of collective sports essentially depends on mutual team members cooperation, and non-cooperation can carry to unsuccessful results [1]. Team games highlight positive participation among players as the strategic basis to achieve match success, and a loss of every player's protagonist role is needed for a team's efficient cooperation strategy [76]. The strategies to organize the actions are indicated by the team's manager regarding each player's profile as well as the specific match circumstance aimed to obtain the most benefit [72]. A fine strategy should include both the individual and the team motivation.

NE mathematical model [81] has been a classic in the design of economy models around the world. In Game Theory, NE is the formal fundament of non-cooperative game and commonly used for decision-making in competitive scenarios [85]. The automation of Baseball strategic gaming by applying the NE for a team selection of strategies is applied and the strength obtained has been analyzed by [3]. However, a NE strategy profile is frequently not Pareto optimal and may not lead to the best decision-making for a team but just to a half-good for individuals, which could, in the long term, have negative impact for the whole team. PE formalism supports the design of Pareto models in economy and, the theoretical optimal benefit. In this thesis with the use of PE and selection of cooperative profiles, we guarantee that the each other commit allows the theoretical optimum on team collaboration. For decades Pareto efficiency has been well known to be a benchmark to select, from a population of solutions, the optimal for a problem in engineering fields, and, in evolutionary algorithms, to select the next generation of individuals[25].

The selection of strategies based on either the PE, or on the NE, or on both, is analyzed. PE rules the team cooperation mind that people's mutual confidence is an assumed condition for a successful team. The abilities of each group member are added in the collective procedure facing a complex task deployment, which allows a theoretical Pareto-efficient design of collective strategies to work up a complex task. However, this theoretical perspective on each member's best strategies, in a real (non-theoretical) match, may not be the times followed. The pass from theory to practice enlightens the usefulness of each of the NE or PE in a real AF match. The relevance of use of each of the equilibriums is shown from the set of computer simulations, which apply either each or both at the opportune moment so to strengthen the team's performance.

For the algorithmic American Football match gaming, we follow the formulation prosed in [3] for Baseball gaming, which formal account is a context-free-grammar like the one for the present American Football algorithmic setting.

3.2 Formal Language

Using the formal CFG (context-free grammar) rules the generation/description of any simple or complex Football gaming, including a whole match, is done. The generated CFG language is read by the associated finite state machine. In order to guarantee the SO of plays in the simulation, likely to human being matches, we use a generator of random numbers such that the number occurrence carries the occurrence of the associated play. In addition, the higher is the SO (frequency of occurrence) of the play the more the chance it is included in the simulation. See details on the flip function in [3]. CFG, finite state machine and the generator of random plays are the algorithmic fundamentals for our automation, which attains similar scores to the human team's matches in real life. By starting with the empty string (ε) each next play is concatenated in a string describing the occurrence of the plays in a match. The super-indexed indicates the player who performs the play.

Let I, I' be different AF teams, $i \in I$ and $i' \in I'$, the CFG for AF follows. Let $\hat{G} = (\Sigma, V - \Sigma, R, B)$ be the CFG:

- V is the alphabet of terminals and non-terminals.
- $\Sigma \subseteq V$ is the set of terminals.
- $V - \Sigma$ is the set of non-terminal elements.
- $R \subseteq (V - \Sigma) \times V^*$ is the set of rules.
- $B \in V - \Sigma$ is the initial symbol.

In a multi-player game, we express the available actions from all the players at the time into the strategy profiles, which are vectors that position i correspond to the player i action, in Table 3.1 is shown the terminals symbols, in Table 3.2 the non-terminal symbols and in Table 3.3 some CFG rules.

Table 3.1 Σ = Terminals symbols.

| Offensive movements | Defensive movements |
|---------------------------------|-----------------------------------|
| kfb^i : kick the ball | tl^i : tackle the player |
| cb^i : catch the ball | sf^i : safety |
| rb^i : run with the ball | ob^i : stop the ball |
| db^i : pass the ball | beo^i : roll back the adversary |
| adb^i : advance with the ball | |
| td^i : touchdown | Penalization movements |
| p^i : punt | h^i : holding |
| ga^i : field goal | fs^i : false start |
| re^i : conversion | dg^i : delay game |
| g^i : goal | |
| s^i : Snap | |

Table 3.2 Non-terminals symbols.

| | |
|--|---|
| B : Initial symbol | RE : Conversion of two point |
| M : Movement after kick off | M_{re} : After conversion of two point |
| M_1 : Movement for catching the ball | M_{re2} : Auxiliary symbol |
| M_2 : Movement for running with the ball | R_{re} : Auxiliary symbol |
| M_3 : Movement for passing the ball | M_{re3} : Auxiliary symbol |
| D_y^{oi} : Denote the downs | P_{rela} : Auxiliary symbol |
| M_5 : Auxiliary symbol | P : Changes team defensive to offensive |
| M_6 : Auxiliary symbol | P_{la} : Auxiliary symbol |
| M_7 : Auxiliary symbol | R : Auxiliary symbol |
| T : Options after touchdown | A_x : yards count |
| PA : Extra point by kicking the ball | |

Table 3.3 $R \subseteq (V - \Sigma) \times V^*$ some grammar rules.

| |
|---|
| $B \rightarrow kfb^i M$: kick off the ball. |
| $M \rightarrow cb^i M_1$: The offensive team catches the ball a make a move. |
| $M_1 \rightarrow rb^i M_2 db^i M_3 {}^j tl^i D_{y=10}^{o1}$: run, or pass the ball, or the player i is tackled by j . |
| $M_2 \rightarrow {}^j tl^i D_{y=10}^{o1} td^i T ob^i D_{y=10}^{o1}$: The player i is tackled by j , or make a touchdown, or the team is stopped. |
| $M_3 \rightarrow cb^i M_1 ob^i D_{y=10}^{o1} cb^{j'} M'_1$: Catch the ball, or the team is stopped, or interception the ball. |
| $D_{y=10}^{o1} \rightarrow D_y^{oi}$: Symbol to define the first down. |
| $D_y^{oi} \rightarrow s^i M_5 p^i M'$: Options in the begging of the down. |
| $D_{y>0}^{o5} \rightarrow D_{y=10}^{o1}$: First down of another team. |

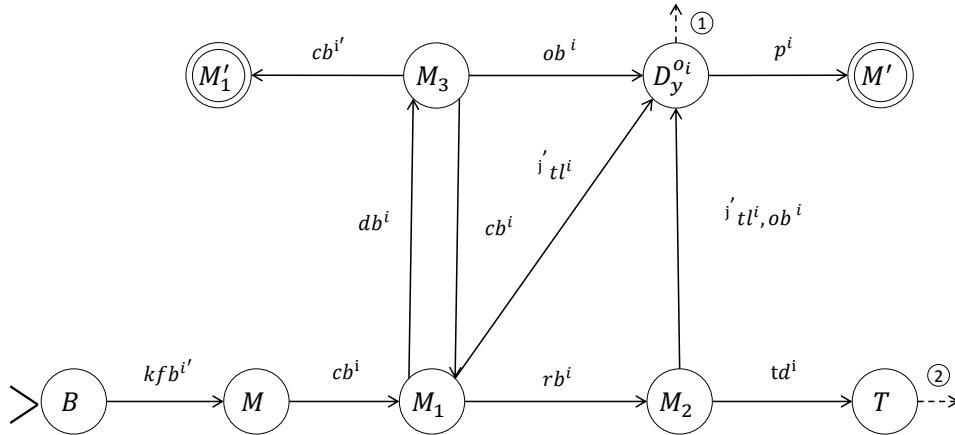
$D_{y \leq 0}^{O_{i < 5}} \rightarrow D_{y=10}^{O_1}$: advance 10 or more yards and get first down.
 $M_5 \rightarrow db^i P | rb^i R | ga^i PA | j^i tl^i A_x | h^i D_{y=y+10}^{O_{i+1}} | fs^i D_{y=y+5}^{O_{i+1}} | dg^i D_{y=y+5}^{O_{i+1}}$: Moves after a play including the penalization ones.
 $T \rightarrow kfb^i PA | db^i RE$: kick off, or pass the ball.
 $PA \rightarrow g^i B' | ob^i B'$: After an extra point, make a goal, or stop the team.
 $RE \rightarrow s^q M_{re}$: Conversion of two points.
 $M_{re} \rightarrow db^i P_{re} | rb^i R_{re} | j^i tl^i B'$: Movements after the conversion of two points.
 $P_{re} \rightarrow cb^i M_{re2} | ob^i B' | cb^{i'} M'_2$: catch the ball, or stop the team.
 $M_{re2} \rightarrow rb^i R_{re} | db^i P_{rela} | j^i tl^i B' | re^i B'$: run, or pass the ball, or the player i is tackled by j , or conversion.
 $R_{re} \rightarrow j^i tl^i B' | re^i B' | ob^i B' | db^i P_{rela}$: The player i is tackled by j , or conversion, or stop the team, or pass the ball.
 $M_{re3} \rightarrow db^i P_{rela} | j^i tl^i B' | od^i B'$: Run, or pass the ball, or the player i tackled by j .
 $P_{rela} \rightarrow cb^i M_{re3} | ob^i B' | cb^{i'} M'_2$: catch the ball, or stop the team.
 $M_6 \rightarrow rb^i R | db^i M_7 | j^i tl^i A_x | ob^i A_x | td^i T$: Run, or the player i is tackled by j , or touchdown.
 $P_{la} \rightarrow rb^i R | db^i M_7 | j^i tl^i A_x | ob^i A_x$: run, or pass the ball, or the player i is tackled by j .
 $M_7 \rightarrow cb^i P_{la} | ob^i A_x | cb^{i'} M'_2 | j^i tl^i A_x$: Actions after kick off by a touchdown.
 $P \rightarrow cb^i M_6 | ob^i D_y^{O_{i+1}} | cb^{i'} M'_2$: M'_2 : Change of team defensive to offensive.
 $R \rightarrow db^i M_7 | j^i tl^i A_x | td^i T | ob^i A_x$: Pass, or the player i is tackled by j , or touchdown, or stop the team.
 $A_x \rightarrow adb_x D_{y=y-x}^{O_{i+1}} | beo_x D_{y=y+x}^{O_{i+1}}$: Sum, or subtraction of yards.

3.3 Finite State Machine

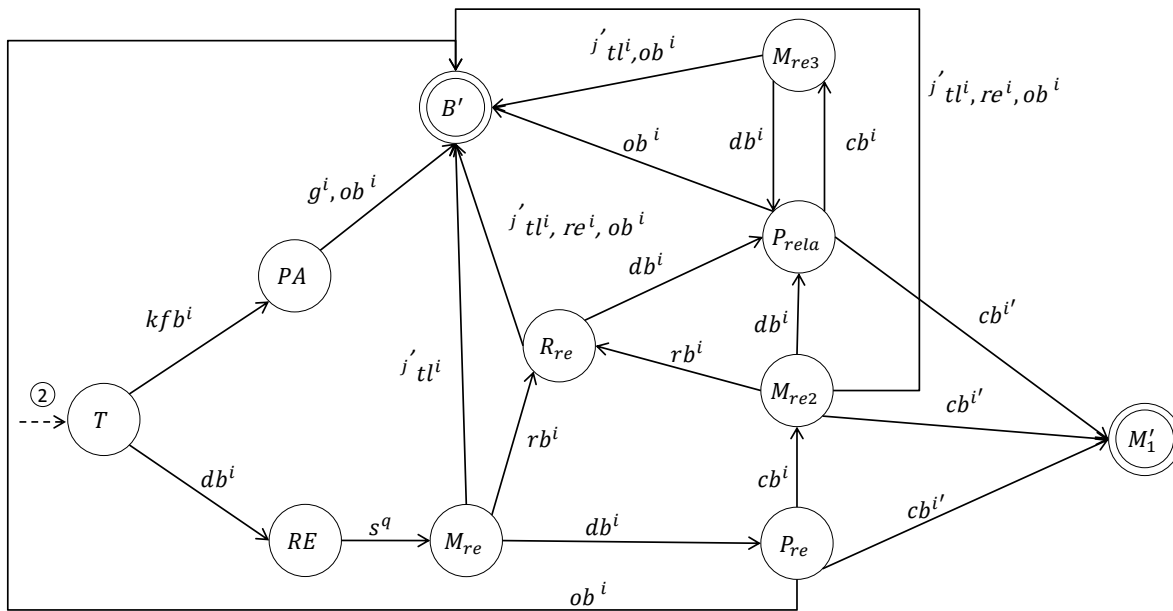
The next finite state machine (FSM) does the algorithmic setting for the AF. Let $(\Sigma, S, s_0, \delta, H)$ be a push-down automata such that:

- Σ is the alphabet.
- $S = \{B, M, M_1, M_2, M_3, D_y^{O_i}, M_5, M_6, M_7, T, PA, RE, M_{re}, P_{re}, M_{re}, M_{re2}, R_{re}, M_{re3}, P_{rela}, P, P_{la}, R, A_x, M'_1, M', B'\}$ is the set of states.
- $\delta: S \times \Sigma \rightarrow S$ is the transitions function.
- $B \in S$ is the initial state.
- $H = \{M'_1, M', B'\} \subseteq S$ is the set of halt states.

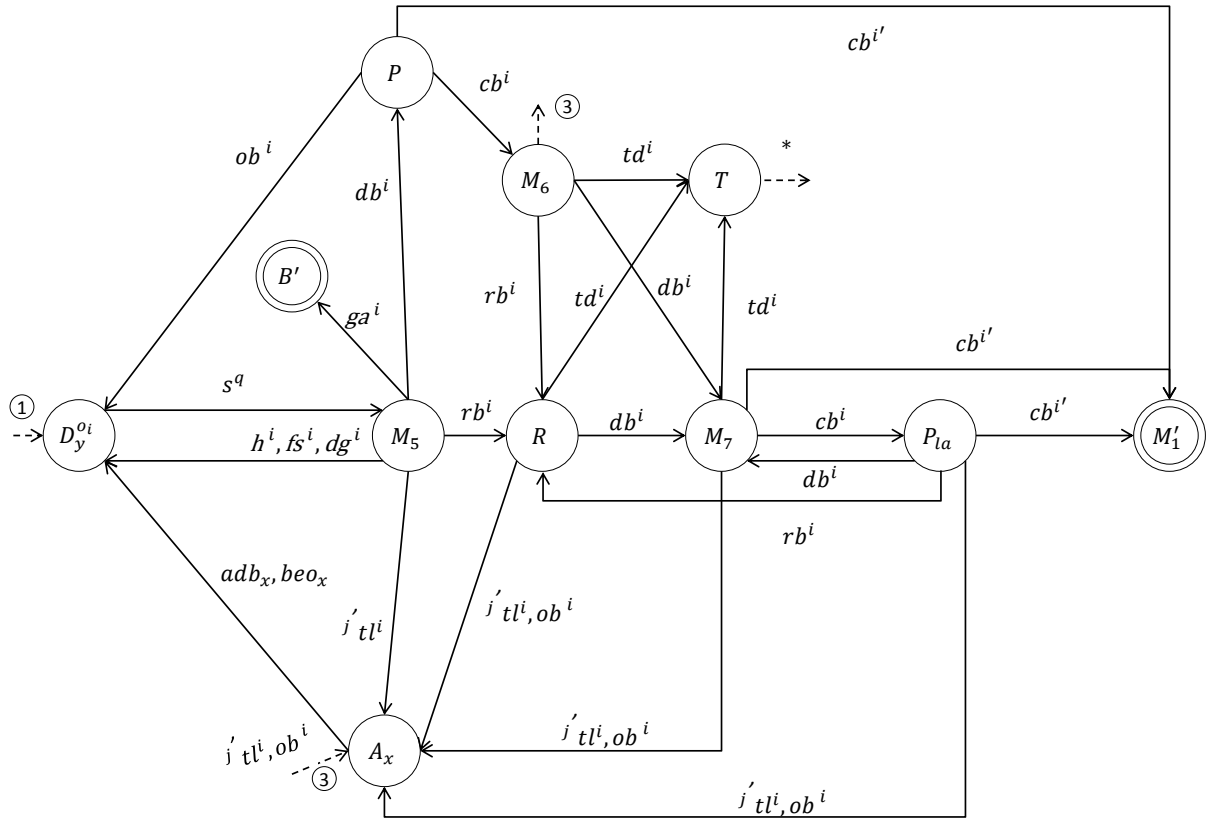
The FSM for (a) the game start, (b) touchdown annotation, and (c) the plays execution in the field is respectively illustrated in Figure 3.1 (a), (b) and (c).



(a) FSM for the game start.



(b) FSM for a *touchdown*.



(c) FSM for the plays description in the field.

Figure 3.1 FSM for AF (a), (b), (c).

These FSMs read the strings that were generated by applying the CFG rules and describe the complex plays during part or a whole American Football match.

3.4 Strategies by Team-Roles and the Average Occurrence of Plays

In this section, we present some AF plays being divided according to the team-role, offensive or defensive (Table 3.4).

Table 3.4 Offensive and defensive plays.

| Off. plays | Description | Def. plays | Description |
|-------------------|--|-------------------|--------------------------|
| <i>kb</i> | Kick the ball | <i>tl</i> | Tackling |
| <i>cb</i> | Catch the ball by product of a pass | <i>sf</i> | Safety |
| <i>rb</i> | Run with the ball | <i>sb</i> | Stop the ball |
| <i>pb</i> | Pass the ball | <i>in</i> | Interception |
| <i>fd</i> | Scoring yards | <i>qs</i> | Tackling the quarterback |
| <i>td</i> | Touchdown | <i>yb</i> | Roll back the contraries |
| <i>p</i> | Extra point (1 point by product of a kick) | <i>fb</i> | Fumble the ball |
| <i>re</i> | Conversion (2 points) | <i>fr</i> | Turnover the ball |
| <i>fg</i> | Field goal | <i>tb</i> | Touchback |

Using real statistical from NFL (National Football League) see http://gametheory.cs.cinvestav.mx/NFL_statistics.pdf, the probability of occurrence of each play above is calculated and listed in descending order in Table 3.5. These values come from performing statistical averages of values in tables showing data by player-role and not by specific player, but for a specific player his individual statistics can be used.

Table 3.5 SO of AF plays.

| Play | Average | $\rho(play)$ | Play | Average | $\rho(play)$ |
|-------------|----------------|--------------|-------------|----------------|--------------|
| <i>sb</i> | 1050.5 | 0.232884067 | <i>p</i> | 39.4375 | 0.008742851 |
| <i>yb</i> | 845 | 0.187327022 | <i>fg</i> | 26.96875 | 0.005978669 |
| <i>tl</i> | 775.6875 | 0.171961218 | <i>td</i> | 25.125 | 0.005569931 |
| <i>pb</i> | 566.75 | 0.125642118 | <i>in</i> | 15.6875 | 0.003477743 |
| <i>rb</i> | 348.484375 | 0.077255077 | <i>fb</i> | 15.09375 | 0.003346115 |

| | | | | | |
|-----------|----------|-------------|-----------|---------|-------------|
| <i>cb</i> | 346.9375 | 0.076912152 | <i>fr</i> | 9.625 | 0.002133755 |
| <i>fd</i> | 319.125 | 0.070746433 | <i>tb</i> | 5.875 | 0.001302422 |
| <i>kb</i> | 78.40625 | 0.017381786 | <i>re</i> | 1.03125 | 0.000228617 |
| <i>qs</i> | 40.46875 | 0.008971468 | <i>sf</i> | 0.625 | 0.000138555 |

3.4.1 Offensive Team Plays

- Offensive linemen players *OL* have two major tasks: 1) block the defensive team members which try to tackle to the quarterback (*QB*), and 2) open ways in order to runners can pass. The *OL* players are, the center, left guard, right guard, left tackle and right tackle. We defined these players as *OL* and the plays to consider are $OL_{plays} = \{tl, yb\}$.
- The quarterback (*QB*) is the offensive leader, whose plays follows, $QB_{plays} = \{rb, pb, fd, td, re, tb\}$.
- The backfield players *BF* are: the halfback, tailback the fullback. The *BF* plays follow, $BF_{plays} = \{rb, fd, td, re, tb, tl\}$.
- Receiver's role *RC* is to catch the ball passed by the *QB*; *RC* players are the tight end and wide. The *RC* plays follow, $RC_{plays} = \{cb, rb, fd, td, re\}$.

3.4.2 Defensive Team Plays

- The defensive linemen players *DL* are: the defensive end, defensive tackle and nose tackle, their main task is to stop running plays on the inside and outside, respectively, to pressure the *QB* on passing plays. The *DL* plays follow, $DL_{plays} = \{tl, sf, sb, qs, yb, fb, fr\}$.
- The linebacker players *LB* have several tasks: defend passes in shortest paths, stop races that have passed the defensive line or on the same line and attack the *QB* plays penetration; they can be three or four. The *LB* plays follow, $LB_{plays} = \{tl, sf, sb, qs, fb, fr\}$.

- The defensive backfield players DS are: the cornerbacks and safeties, which major task is to cover the receivers. The DS plays follow, $DS_{\text{plays}} = \{tl, in, fb, fr\}$.

3.4.3 Special Team Plays

- Kicker player K kicks off the ball and do field goals and extra points. The kicker's plays follow, $K_{\text{plays}} = \{kb, p, fg\}$.
- The kickoff returner R is the player on the receiving team who catches the ball. The plays are $R_{\text{plays}} = \{rb, td, tb\}$.

3.5 Setting-Up of Payoff Functions

The each role's payoff function to value the strategy profiles, selects the own convenience value by regarding:

- For QB is important to make a pass, his characteristic move, even with a touchdown scoring can generate a greater personal gain.
- The basic action of RC is to increase the score, but to make it happens he must catch the ball and run to the touchdown line.
- The OL main function is tackling the adversary to allow QB send pass; as well, open space for RC ball runs, or, in some cases, push back the opposing team.
- The BF preferred score is touchdown or conversion, and should run to get there. Other option is to get a first down, or tackling a player of the opposing team.
- The DL should be tackling the opposing QB , roll back yards to the opposing team or get a safety; in descent order of importance the following is to stop the ball, tackling and cause fumbles and try to recover it by the opponent.
- The main function of LB is to recover a lost ball and then could be to generate a safety.
- For DS intercepting a pass would be best, but it is also important to get the other team loses control of the ball.

- For K , the most important is to make a field goal, followed by an extra point and typically perform the corresponding kicks.
- For R , the best choice is to score a touchdown with the return of the kick, but usually just run until stopped, or perform touchback for time.

We propose that the player-roles' skills are qualified on the base of the player-roles' performance on certain plays, and the statistics resumes these qualifications. Let $u_i(x_1, \dots, x_i, \dots, x_n) = V_1(x_1) * p(x_1) + \dots + V_i(x_i) * p(x_i) + \dots + V_n(x_n) * p(x_n)$ be the payoff function of the player-role i , $(x_1, \dots, x_i, \dots, x_n)$ is a strategy profile such that x_i is one play of player-role i , The factors in the payoff function are: $V_i(x_i)$ represents the player-role i 's preference on the play x_i , and $p(x_i)$ is the average statistics of the player-role on play x_i , by regarding the NFL statistics [83]; as well, the other elements in the formula are the contributions of the other player-roles whom directly share the play.

3.5.1 Offensive Team

Let define the strategy profile for offensive team as (w, x, y, z) , with $w \in QB_{\text{plays}}$, $x \in RC_{\text{plays}}$, $y \in OL_{\text{plays}}$, $z \in BF_{\text{plays}}$.

- For QB , we should consider the QB plays as well as the OL plays, the payoff function (3.1) follows.

$$u_{QB}(w, x, y, z) = V_{QB}(w) * p(w) + V_{OL}(y) * p(y) \quad (3.1)$$

- For RC , we should consider the RC plays, the QB as well as the OL plays, the payoff function (3.2) follows.

$$u_{RC}(w, x, y, z) = V_{RC}(x) * p(x) + V_{QB}(w) * p(w) + V_{OL}(y) * p(y) \quad (3.2)$$

- For *BF*, we should consider the *BF* plays, the *QB* plays as well as the *OL* plays, the payoff function (3.3) follows.

$$u_{BF}(w, x, y, z) = V_{BF}(z) * p(z) + V_{QB}(w) * p(w) + V_{OL}(y) * p(y) \quad (3.3)$$

- For *OL*, we should only consider the *OL* plays, the payoff function (3.4) follows.

$$u_{OL}(w, x, y, z) = V_{OL}(y) * p(y) \quad (3.4)$$

3.5.2 Defensive Team

Let define the strategy profile for defensive team as (x, y, z) where $x \in DL_{plays}$, $y \in LB_{plays}$, $z \in DS_{plays}$.

- For *DL* and *LB*, we should consider *DL* plays as well as *LB*plays, the payoff function (3.5) follows.

$$u_{DL|LB}(x, y, z) = V_{DL}(x) * p(x) + V_{LB}(y) * p(y) \quad (3.5)$$

- For *DS*, we should only consider the *DS* plays, the payoff function (3.6) follows.

$$u_{DS}(x, y, z) = V_{DS}(z) * p(z) \quad (3.6)$$

3.5.3 Special Team

- For *K*, the payoff function (3.7) follows.

$$u_K(x) = V_K(x) * p(x) \text{ where } x \in K_{plays} \quad (3.7)$$

- For *R*, the payoff function (3.8) follows.

$$u_R(x) = V_R(x) * p(x) \text{ where } x \in R_{plays} \quad (3.8)$$

3.5.4 Examples

We use the set of values in Tables 3.6 – 3.7 that are assigned according to each player's preference values on each of the own plays, and are used to calculate the payoff functions, that in turns are used to find out the strategy profiles that fit the Nash equilibrium condition.

Table 3.6 Values of offensive plays by player.

| <i>QB</i> | <i>RC</i> | <i>OL</i> | <i>BF</i> |
|--------------------|--------------------|--------------------|--------------------|
| $V_{QB}(rb) = 0.5$ | $V_{RC}(cb) = 0.7$ | $V_{OL}(tl) = 0.5$ | $V_{BF}(rb) = 0.7$ |
| $V_{QB}(pb) = 0.7$ | $V_{RC}(rb) = 0.7$ | $V_{OL}(yb) = 0.6$ | $V_{BF}(fd) = 0.4$ |
| $V_{QB}(fd) = 0.6$ | $V_{RC}(fd) = 0.6$ | | $V_{BF}(td) = 0.9$ |
| $V_{QB}(td) = 0.8$ | $V_{RC}(td) = 0.9$ | | $V_{BF}(re) = 0.8$ |
| $V_{QB}(re) = 0.9$ | $V_{RC}(re) = 0.8$ | | $V_{BF}(tl) = 0.5$ |
| $V_{QB}(tb) = 0.5$ | | | |

Table 3.7 Values of defensive plays by player.

| <i>DL</i> | <i>LB</i> | <i>DS</i> |
|--------------------|--------------------|--------------------|
| $V_{DL}(tl) = 0.5$ | $V_{LB}(tl) = 0.5$ | $V_{DS}(tl) = 0.5$ |
| $V_{DL}(sf) = 0.8$ | $V_{LB}(sf) = 0.8$ | $V_{DS}(in) = 0.9$ |
| $V_{DL}(sb) = 0.6$ | $V_{LB}(sb) = 0.6$ | $V_{DS}(fb) = 0.7$ |
| $V_{DL}(qs) = 0.9$ | $V_{LB}(qs) = 0.9$ | $V_{DS}(fr) = 0.8$ |
| $V_{DL}(yb) = 0.8$ | $V_{LB}(fb) = 0.7$ | |
| $V_{DL}(fb) = 0.5$ | $V_{LB}(fr) = 0.8$ | |

Now, we define the set of strategy profiles. The set of strategy profiles for offensive team, (w, x, y, z) where $w \in QB_{\text{plays}}, x \in RC_{\text{plays}}, y \in OL_{\text{plays}}, z \in BF_{\text{plays}}$ is $OffensiveT_{\text{profiles}} = \{(rb, rb, tl, rb), (rb, rb, tl, fd), \dots, (tb, re, yb, tl)\}$. The set of strategy

profiles for defensive team, (x, y, z) where $x \in DL_{\text{plays}}, y \in LB_{\text{plays}}, z \in DS_{\text{plays}}$ is $DefensiveT_{\text{profiles}} = \{(tl, tl, tl), (tl, tl, in), \dots, (fr, fr, fr)\}$. Using the payoff functions defined in Section 3, each strategy profile is valued by respective player's payoff function. Some illustrative examples follow.

Offensive team

- For QB , $u_{QB}(rb, rb, tl, rb) = V_{QB}(rb) * p(rb) + V_{OL}(tl) * p(tl)$ is the payoff function that only embrace QB and OL plays. Reason is that RC and BF plays do not relevant impact the QB plays, so not the QB payoff function valuations. $u_{QB}(rb, rb, tl, rb) = 0.5 * 0.077255077 + 0.5 * 0.171961218$. $u_{QB}(rb, rb, tl, rb) = 0.124608$.
- For RC , $u_{RC}(rb, rb, tl, rb) = V_{RC}(rb) * p(rb) + V_{QB}(rb) * p(rb) + V_{OL}(tl) * p(tl)$ is the payoff function, that only embrace RC , QB and OL plays. Reason is that the BF plays do not relevant impact the RC plays, so not the RC payoff function valuations. $u_{RC}(rb, rb, tl, rb) = 0.7 * 0.077255077 + 0.5 * 0.077255077 + 0.5 * 0.171961218$. $u_{RC}(rb, rb, tl, rb) = 0.1786867014$.
- For BF , $u_{BF}(rb, rb, tl, rb) = V_{BF}(rb) * p(rb) + V_{QB}(rb) * p(rb) + V_{OL}(tl) * p(tl)$ is the payoff function, that only embrace BF , QB and OL plays. Reason is that the RC plays do not relevant impact the BF plays, so not the BF payoff function valuations. $u_{BF}(rb, rb, tl, rb) = 0.7 * 0.077255077 + 0.5 * 0.077255077 + 0.5 * 0.171961218$. $u_{BF}(rb, rb, tl, rb) = 0.1786867014$.
- For OL , $u_{OL}(rb, rb, tl, rb) = V_{OL}(rb) * p(rb)$ is the payoff function, that only embrace the OL plays. $u_{OL}(rb, rb, tl, rb) = 0.5 * 0.171961218$. $u_{OL}(rb, rb, tl, rb) = 0.085980609$.

Defensive team

- For DL and LB , $u_{DL|LB}(tl, tl, tl) = V_{DL}(tl) * p(tl) + V_{LB}(tl) * p(tl)$ is the payoff function, that only embrace the DL and LB plays. Reason is that the DS plays do not relevant impact the DL and LB plays, so not the DL and LB payoff function

valuations. $u_{DL|LB}(tl, tl, tl) = 0.5 * 0.171961218 + 0.5 * 0.171961218.$

$u_{DL|LB}(tl, tl, tl) = 0.171961218.$

- For DS , $u_{DS}(tl, tl, tl) = V_{DS}(tl) * p(tl)$ is the payoff function, that only embrace the DS plays. $u_{DS}(tl, tl, tl) = 0.5 * 0.171961218.$ $u_{DS}(tl, tl, tl) = 0.085980609.$

We should calculate all payoff values on strategy profiles using the players' payoff functions. For the offensive team the strategy profile (pb, rb, yb, tl) satisfies the Nash equilibrium condition, and for the defensive team, the NE profile is (yb, sb, tl) . So, the best combination of offensive plays is by combining a pass from QB , the RC running with the ball and the OL and BF opening the way and stopping their opponents. Notice that the Nash equilibrium strategy profile depends on the player's preference value as well as on the player's payoff function. For more examples with different preference values and payoff functions, please visit http://gametheory.cs.cinvestav.mx/Examples_AF_Analysis_of_Strategies.pdf.

3.6 Selection of Strategies: Merging of and Experiments

Experiments concern the performance comparison of teams that use a method for selection of strategies with regard to the next match gaming conditions:

- Comparing the NFL results from some teams against the simulation results by applying NE or PE.
- Comparing NE versus PE in teams with the same playing characteristics (NFL statistics).

Analysis of results is also given.

3.6.1 Simulations using NFL Data or Selection of Strategies

To simulate the players' actions according to their performance, we use NFL real statistics from the Denver Broncos (DEN) and Oakland Raiders (OAK¹) in the 2012 season.

Using the NFL statistics, the SO of each AF play is used to induce the SO of the play can happen in a computer match simulation. Next, we do a comparison among simulations of AF matches using NFL statistics, without any concern for analysis of strategies, versus simulations that use NE or PE as the methods for selection of strategies. One thousand computer simulations per each of the next conditions were carried out.

- 1) Team 1 (T_1) uses DEN statistics versus Team 2 (T_2) uses OAK¹ statistics.
- 2) T_1 uses DEN statistics versus T_2 uses OAK¹ statistics and NE as strategic analysis.
- 3) T_1 uses DEN statistics versus T_2 uses OAK¹ statistics and PE as strategic analysis.

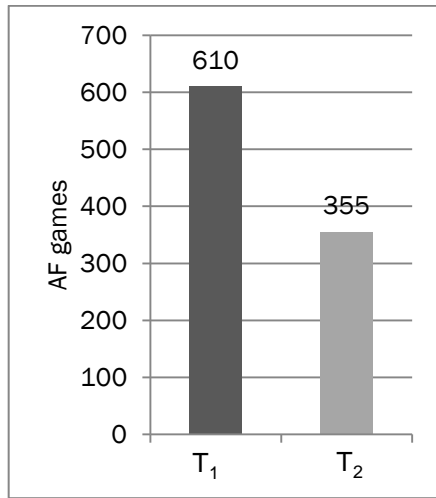


Figure 3.2 DEN stats versus OAK¹ stats.

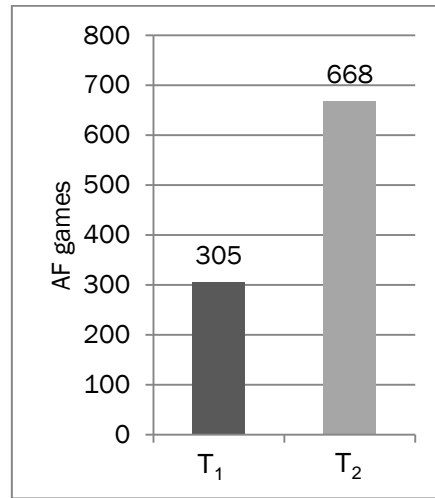


Figure 3.3 DEN stats versus OAK¹ stats and NE.

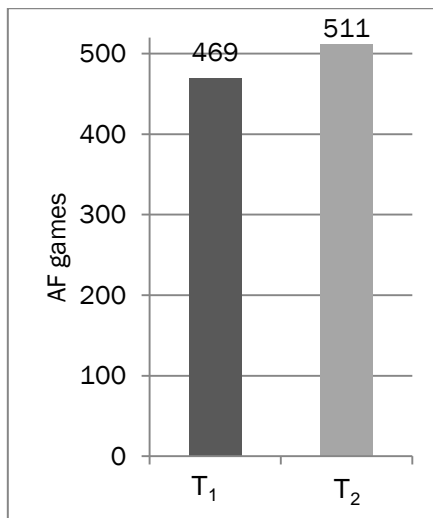


Figure 3.4 DEN stats versus OAK¹ stats and PE.

By considering the results in Figure 3.2, when T₁ uses DEN statistics and T₂ uses OAK¹ statistics, T₁ is superior, 610/355 wins respectively. Figure 3.3 shows when T₁ uses DEN statistics and T₂ uses OAK¹ statistics and NE, and T₂ is superior, 305/668 wins respectively. Figure 3.4 shows when T₁ uses DEN statistics and T₂ uses OAK¹ statistics and PE, and 469/511 wins in favor of T₂.

3.6.2 Using Nash and Pareto Efficiency in Teams with Common NFL Statistics

In this section of tests, we highlight the use of NE and PE in teams with same playing characteristics (same statistics), in order to measure the performance of both techniques. One hundred computer simulations of AF matches where both teams, T_1 and T_2 use DEN statistics were carried out per each of the next conditions.

- 1) T_1 uses NE and T_2 only uses statistics.
- 2) T_1 uses PE and T_2 only uses statistics.
- 3) T_1 uses NE and T_2 uses NE.
- 4) T_1 uses NE and T_2 uses PE.
- 5) T_1 uses PE and T_2 uses NE.
- 6) T_1 uses PE and T_2 uses PE.

Considering the results in Figures 3.5 – 3.6 (item 1-2), T_1 uses NE or PE for selection of strategies while T_2 only uses statistics in AF gaming, T_1 has advantage over the T_2 . In Figure 3.7 (items 3), T_1 and T_2 remain balance 477/472 both use NE. In Figure 3.8 (items 4), T_1 uses NE and it is superior to T_2 that uses PE, 681/259 wins respectively. In Figure 3.9 (items 5), T_1 uses PE versus T_2 that uses NE, T_2 is superior 302/622 and In Figure 3.10 (items 6), both teams use PE and both remain balance 481/471.

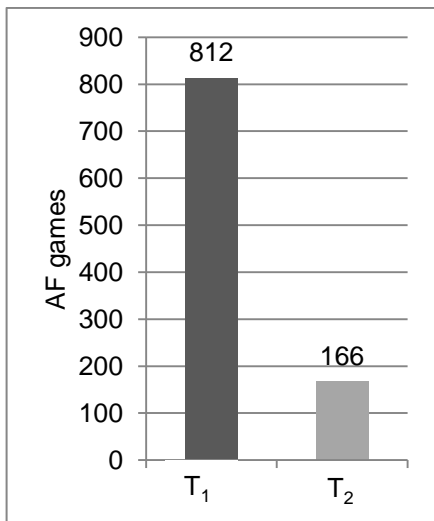


Figure 3.5 NE versus stats.

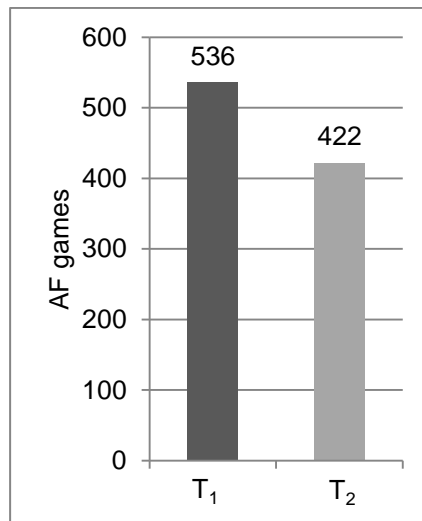


Figure 3.6 PE versus stats.

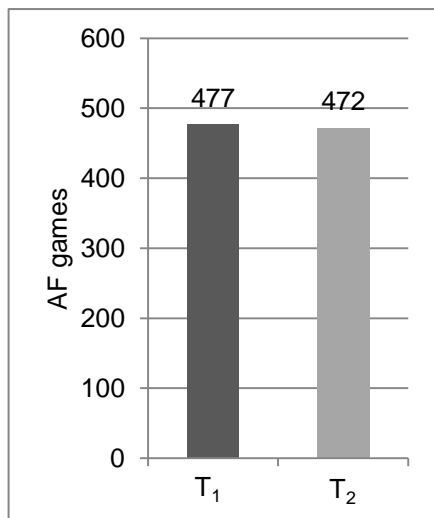


Figure 3.7 NE versus NE.

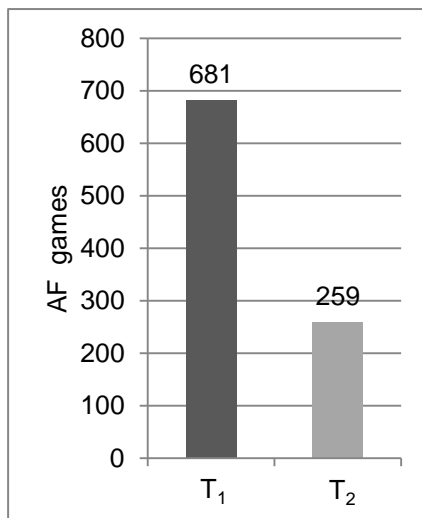


Figure 3.8 NE versus PE.

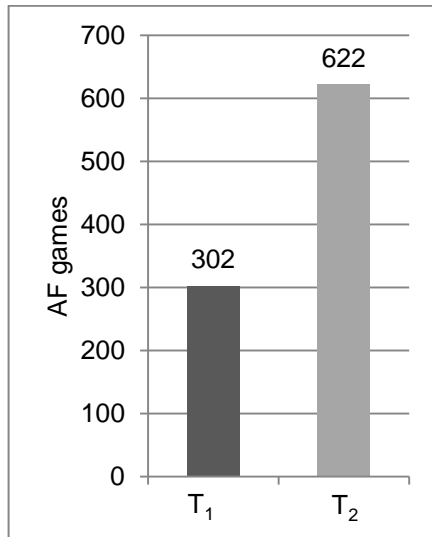


Figure 3.9 PE versus NE.

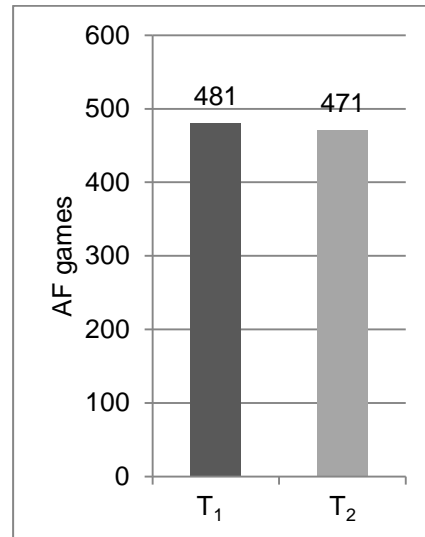


Figure 3.10 PE versus PE.

3.6.3 Analysis of Results

Next, analysis focuses on the behavior when both teams T_1 and T_2 , use different techniques. Figure 3.11 illustrates when T_1 only uses DEN statistics while T_2 uses OAK¹ statistics, or PE, or NE. Statistically, when T_1 uses DEN statistics and T_2 OAK¹ statistics, T_1 has a superior performance because in the NFL 2012 season, DEN team had a superior performance than OAK¹ team but when T_2 uses either PE or NE for selection of strategies his performance increased even overcome T_1 . In this case, we emphasize that with the use of NE or PE for selection of strategies. The teams increased their playing level, selecting the most appropriate strategies given the present AF circumstances. Even if the teams are inferior statistically to their opponents.

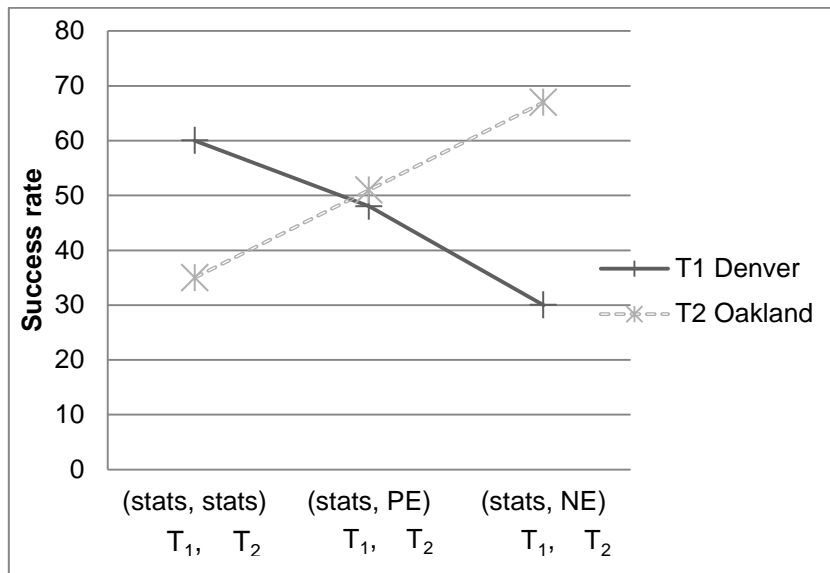


Figure 3.11 T₁ only uses stats and T₂ uses stats, PE and NE.

In the following figures (Figures 3.12 – 3.14) are shown comparisons regarding when teams have the same playing characteristics, showing how the team performance is changed according with the method of selection of strategies used. Figure 3.12 illustrates when T₁ only uses DEN statistics while T₂ uses DEN statistics, or PE, or NE. In this case, since T₁ remains with DEN statistics and T₂ changes his strategy analysis, T₂ performance is improved. Figure 3.13 shows when T₁ only uses PE while T₂ uses DEN statistics, or PE, or NE. In this case, T₁ performance is worse only when T₂ uses NE for selection of strategies. And Figure 3.14 illustrates when T₁ only uses NE while T₂ uses DEN statistics, or PE, or NE. T₁ and T₂ performance remains balance until both use NE.

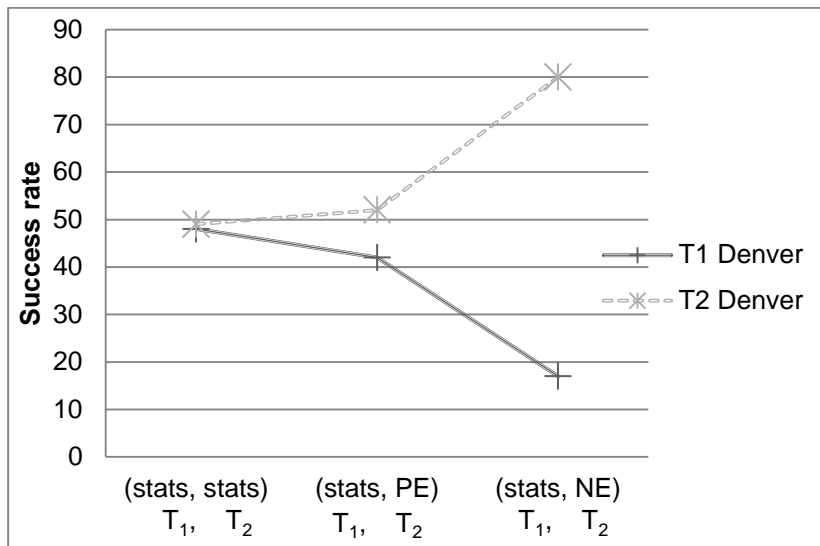


Figure 3.12 T₁ only uses stats and T₂ uses stats, PE and NE.

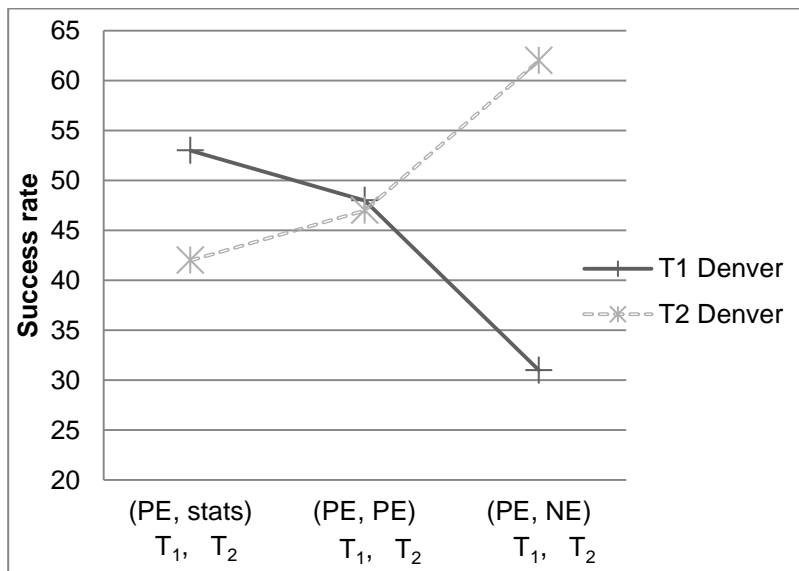


Figure 3.13 T₁ only uses PE and T₂ uses stats, PE and NE.

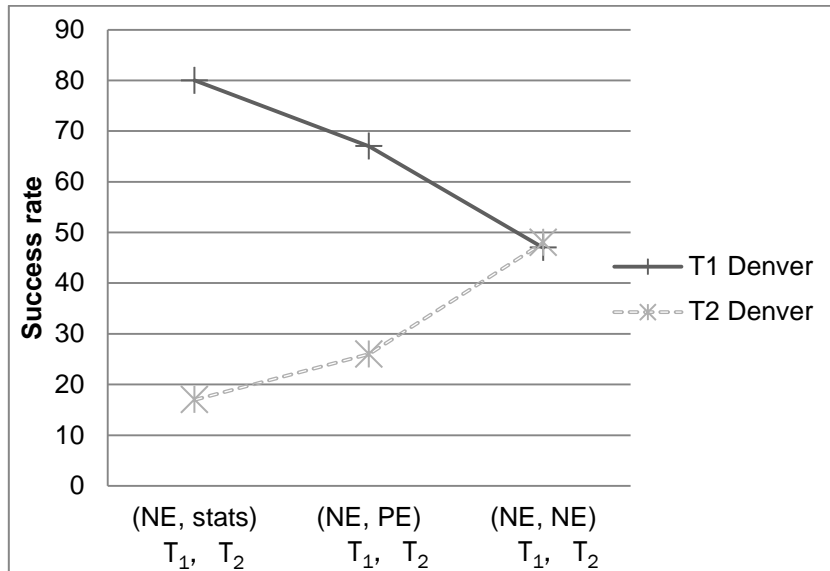


Figure 3.14 T₁ only uses NE and T₂ uses (stats, PE and NE).

3.6.4 Score Forecasting

Our approach can be used to forecast the results of team games and predict the exact scores. Reliable and realistic results are obtained from the computer simulations of AF games using a formal language, a FSM and a generator for American Football plays (see the formal description in Section 2). Within our approach, all of the possible ways to play AF are considered from the start to the end of a game: real games among NFL teams are simulated by basing all of the players' actions on their own NFL statistics. In particular, the complex scoring plays presented by Baker and McHale [8] are included in our model, as given below.

- A touchdown with kickoff return: T₂ kicks the ball, and the kick returner from T₁ scores a touchdown:

$$kfb^4team2\ cb^9rb^9td^9.$$

- A touchdown with a one-point conversion: the quarterback makes two passes to score a touchdown, followed by a one-point conversion:

$$s^{qb}\ db^1\ cb^1\ 1tl^{12}\ s^{qb}\ db^4\ cb^4\ rb^4td^4\ s^{10}re^1.$$

- A touchdown with a two-point conversion: the quarterback makes two passes to score a touchdown, followed by a two-point conversion:

$$s^{qb} db^1 cb^1 tl^{12} s^{qb} db^4 cb^4 rb^4 td^4 s^{qb} db^2 cb^2 re^2.$$

- A safety, i.e., a ball carrier is tackled in his own end zone: T₂ kicks the ball, and the kick returner 1 of T₁ is tackled in his own end zone by player 6:

$$kfb^3 team2 cb^1 tl^6 sf^1.$$

- A field goal: after three plays towards the opponent's end zone, the team decides to kick a field goal:

$$s^{qb} fs^1 s^{qb} db^4 cb^4 tl^{12} \dots s^{10} ga^2.$$

The aforementioned strings describe particular routes to score points, although there are other routes are possible. Recall that to perform one part of the experiments described in the previous section, one hundred computer simulations are conducted on games between the Denver team and the Oakland team using only the NFL statistics for the 2012 season and without making any strategic choices. The winning percentage and the average points that are obtained in one hundred computer simulations are reported and compared with the real scores for the games in the 2012 season (**Table 3.8**). The results show a high degree of accuracy for the forecasting of the exact scores, i.e., there is a difference of ± 1.21 points between the actual and predicted scores.

:

Table 3.8 Forecasting game results using computer simulations

| Team | Winning percentage | Average points | Actual score |
|---------|--------------------|----------------|--------------|
| Denver | 62% | 27.21 | 26 |
| Oakland | 38% | 12.25 | 13 |

Song et al. [104] have stated that statistical models may yield more accurate forecasts than human judgment because objective criteria are employed in models to guard against bias and the non-rational interpretation of data. However, statistical models

sometimes cannot capture non-quantitative factors; hence, forecasts are not completely accurate. Our model produces a high precision for forecasting winning teams and exact scores.

Baker and McHale [8] used a forecasting model with a continuous-time Markov birth process to analyze the ways in which points could be scored in NFL games. The authors focused on an unconverted touchdown (6 points), a touchdown with a one-point conversion (7 points), a touchdown with a two-point conversion (8 points), a safety (2 points), and a field goal (3 points). For each type of score, various hazard functions were used for each team, home and away, that depended on the state of play. As previously described, our developed approach can be used to formally score these particular circumstances by substituting a probabilistic generator for the hazard functions and finite state automata for the Markov process.

3.7 Discussion

By using NFL statistics our proposed approach can forecast the result of matches of teams which never played among them; moreover, the computer simulation results obtained are reliable and realistic since all players' actions are simulated based on their own statistics and hence we can analyze players' behavior under certain circumstances in the match. Generalizes of [8, 52, 104] follows. Besides, the analysis of strategies can be introduced for a whole simulation of a match. For selecting a strategy we can apply specific strategies for the different match circumstances. Decision making for gaming can be, by using the statistics, or by using a kind of the strategies machine based on the NE or the PE.

In our approach the computer simulations of American Football matches are simple and correct. The context-free-language guaranties the correctness of the computer simulations. The SO of the plays during the human being matches, taken from NFL statistics, guaranties the real simulation results. Although Baseball and AF are largely different in how to play and the game rules, the formal modeling and the algorithmic setting of both games are similar because the next reason: Baseball and AF are multi-player sport

games, but each gamer having specific roles that do following ordered strategies during the offensive and defensive steps, as one member of a team obeying the manager. Hence, the formal modeling of AF game is quite similar as the one of Baseball; besides, the essential use of statistics for strategic analysis in both sports is a determining fact for a right decision making that should be regard during match gaming simulations.

A successful analysis of strategies in both gaming matches is supported by the use of NE and PE that are relevant tools for the selection of the players' proper actions during a match, and help to increase the expectations of team success. This partial analysis on strategies for AF match gaming can be improved by following specific challenges: on principal is a whole exploration in all the downs and apply specifics ways to play depending on the adversaries.

In the perspective of multi-agent systems, the authors proposed an Iterated Cooperative Equilibrium (ICE) [23]. In each round the players forecast how the game would be played if they form coalitions, and select their actions accordingly; up to the reward to be obtained the participants' behavior change and the Nash equilibrium convergence is not mandatory, but cooperation behavior can be observed. Tests are practiced with the prisoner's dilemma, the traveler's dilemma and the public goods game. American Football is a team sport game where individualities are meaningful as well. Team cooperation and individualism are equally relevant in an AF match. In our analysis, a meaningful fact is that Nash equilibrium is used to identify relevant circumstance of cooperation in an AF match. When some players should sacrifice their ambitious to ensure a better team result: theoretical best actions, touchdown by long ball pass, is low probably to occur so give a major chance to more probably play, step by step ball carrying, is need. The proposed algorithm for Nash equilibrium on American Football, has a low computational complexity k^n , with k is the number of strategy profiles and n is the number of players. In the worst case, k is around 2000 strategy profiles and n is 11; therefore, the computer complexity is polynomial time.

In the context of an AF match, Pareto efficiency identifies the best actions for the whole team, beyond their plausibility of occurrence. Nash equilibrium can be used to identify

team actions with more realistic plausibility of occurrence. Cooperation passes by the players' ambitious sacrifice to practice a more probably play.

Dual equilibrium (DE) with respect to NE for two players is studied in the so called prescriptive games, Corley et al. [29]. In DE each player acts motivated by the others' best interest and non-selfish behavior influence the outcomes. The essential concept in DE correspond the cooperation in PE since both equilibriums formalize the theoretically optimal team collaboration; PE is a generalized formal account of DE. The altruism and envy behavior in contests for two players is formally analyzed by Kai Konrad [65].

Share in outcomes, at which altruists and envious players have identical payoffs in the games are observed; Konrad claims that the presence of altruism and envy behavior provide stability to the whole population dynamic. We emphasize the relevance of both, cooperation and non-cooperation behavior in human relationships. In our AF analysis both attitudes cooperation and non-cooperation, result in a complementary advantage for the team. We remark again that in some circumstances of AF match, effective cooperation can be identified by mean of the use of NE, beyond the shared strategy profiles that fit both NE and PE.

The design and use of collective strategies has an impact far beyond the field of multi-player sports. Dornhaus [39] analyzed the behavior of social insects, such as ants and bees, and showed that individual-based models can be used to identify non-intuitive benefits of different mechanisms of communication and division of labor. Dornhaus also found that these benefits may depend on the external environment and concluded that individual-based models are useful for testing hypotheses about the benefits of different collective strategies under varying ecological conditions. Roy [97] studied collective strategies in businesses to define the conditions under which this type of strategy can emerge and stabilize and demonstrated the endogenous nature of the dissolution of the strategy.

Coen in [26] studies multiple-team social dilemma integrating the empirical studies of how the people behave and how the people should behave by simulation. It does so by examining the findings of each separate approach to the single team social dilemma, then it applies elements of each approach to the multiple-team social dilemma. This ap-

proach first induces multiple decision rules from empirical outcomes. Second, it analyzes which of these rules-in-use in the multiple-team context provides the best outcome for the individuals involved. i.e., the empirical studies attend the how people make decision and the computer simulations explore which decisions are more effective. Our approach has similarities on this study [26] since we proposed the use of two methods for the analysis of strategies according to different circumstances of AF match for exploring the different players' actions in order to choose the most appropriate team strategy to increase the team performance and also the use of computer simulations is for measuring team strategies using either real statistics of NFL, or NE, or PE.

3.8 Conclusion

American Football is one of the top sport games, the analysis of strategies is essential for a team success. In this Chapter was presented a formal modeling of American Football using a formal grammar and finite state machine. The NFL statistics are used for obtaining realistic computer simulations of AF matches as the matches of NFL teams. The manager's decisions critically influence the gameplay, such that one bad decision can result in poor team performance. A decision based on analytical strategic methods, such as the NE and/or PE, strengthens the team performance, thereby increasing the expectations of winning. The results of computer simulations showed that using the NE for strategy selection improved the team performance over using the PE, even though the PE fits the theoretical Pareto-efficient selection of the strategy profiles, thereby incorporating each member's best strategies. However, in a real (non-theoretical) match, these strategies are unlikely to occur and are therefore impractical. Thus, application to practical situations illustrates the relative utility of the NE and the PE for strategy selection in real AF matches. Applying these methods to strategy selection in a multi-player game can provide the team manager with information to make more successful decisions. Beyond its application to strategic analysis in AF, the developed mathematical model and algorithmic simulation can also be used to analyze topics that involve multi-

agent contributions and interactions among a complex group of entities' actions for which a set of rules is followed to achieve well-defined objectives.

4

The Game of GO

The formal analysis of the board game called “Go” is at the core of advances in computer science in the same way the analysis of Chess was during the 20th century [71]. Go is a top complex board game and currently, design and implement learning methods for Go gaming automation are central challenges for computational intelligence, to demonstrate sufficient skill to beat the top human Go masters.

The **basic tactics** of Go of eyes, ladders and nets are used to dominate a local area [110] (see Figure 4.1). An *eye* is a single empty point enclosed by stones of the same color, which cannot be occupied by an adversary’s stone owing to the suicide rule. Two eyes inside a stone make its capture very difficult. A stone having only one liberty is in *Atari*. A *ladder* results from a sequence of moves that forces an adversary’s stone into atari. A *net* is a set of stones (not always a chain) that surrounds an adversary’s stone such that it could eventually be captured [64]. All these *a-priori*-known patterns of Go basic tactics should be recognized for a fair Go game, and are used for training the NN in learning their recognition.

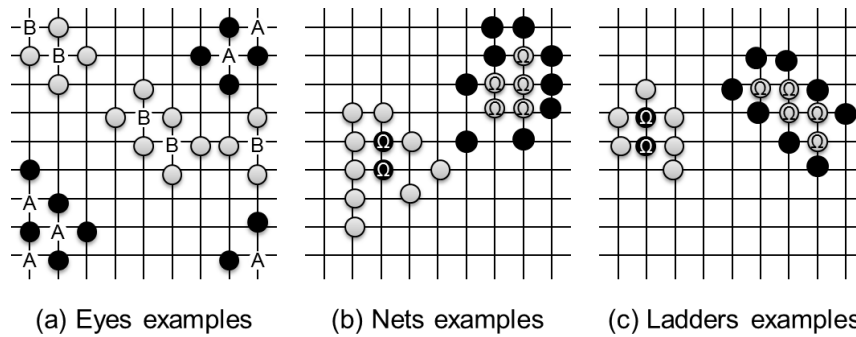


Figure 4.1 Go basic tactics: (a) eyes, unavailable points – A is black and B is white; (b) Ω stones are surrounded by a net and may soon be captured; (c) Ω stones are in atari by ladders.

For broad territory control, Go **strategies** follow a set of planned actions, deployed partly by using the aforementioned tactics as elements. Basic strategies are invasion, reduction, connection, and capture [110] (see Figure 4.2). An *invasion* strategy places a stone near friendly stones, in an area where the adversary's stones look likely to dominate. A *reduction* strategy places a stone near friendly stones, to connect them if needed, in an area likely to be occupied eventually by the adversary. *Capture* reduces the liberties of an adversary's stone to zero and removes it from the board.

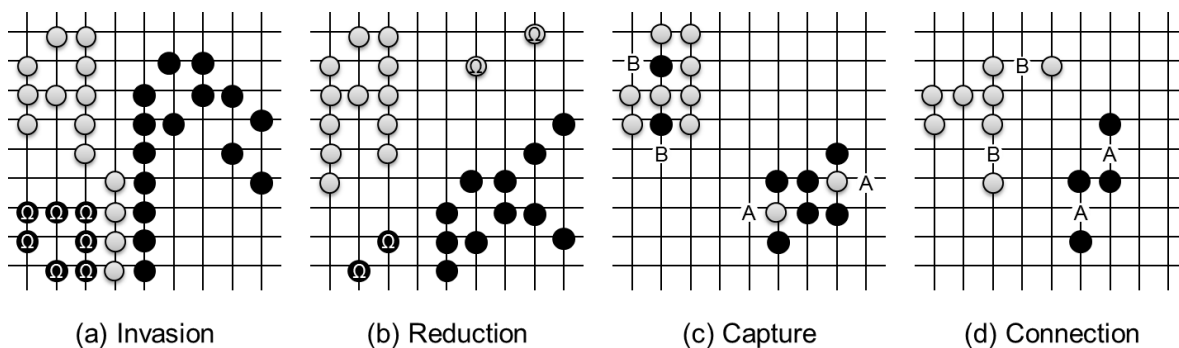


Figure 4.2 Go basic strategies: (a) Ω stones perform invasion in territory dominated by white; (b) Ω black/white stones perform reduction in territory of white/black dominance; (c) black/white playing in positions A/B capture white/black stones; (d) black/white playing in positions A/B perform connections with friendly stones.

A Go gaming strategy move, from the root node to the leaves nodes, is aiming to win a match efficiently. Despite the disarming simplicity of the Go rules, Go gaming conceals a huge combinatorial complexity [6, 43] (see Table 4.1) and therefore, the big complexity is to set up an efficient strategy for playing Go. The **state space complexity** is the number of all the possible arrangements of the game board, which in a 19 × 19 board is about $3^{19 \times 19} \approx 10^{172.24}$ for Go, whereas it is 10^{50} for Chess and 10^{18} for checkers. The branching factor for Go ranges from 200–300 possible moves at each player’s turn; for Chess, the range is 35–40 moves. The **game tree size** is the total number of different matches that can be played and for Go that is $\approx 10^{360}$ (chess $\approx 10^{123}$ and checkers $\approx 10^{54}$). Even on the 9 × 9 board size, the state space and the game tree size is astronomically large [40].

Table 4.1 Complexity of Go, Chess and Checkers games

| Game | Board size | State space | Game tree size |
|-------------|-------------------|--------------------|-----------------------|
| Go | 19 x 19 | 10^{172} | 10^{360} |
| chess | 8 x 8 | 10^{50} | 10^{123} |
| Checkers | 8 x 8 | 10^{18} | 10^{54} |

4.1 State of the Art

Nowadays, the playing level of the best current Go automated simulator versus human being is still modest compared to the great successes achieved in other games of skill such as chess [78]. Current best Go automated players mostly use MCTS and Artificial Neural Networks (NN). The main focus for Go automation is on evaluating non-final positions for estimating the potential on territory occupy [13, 92, 105]. Prospective methods for programming Go gaming are being deployed in AI related domains like simulation-based search algorithm, evaluation functions, heuristic search, machine learning and automatic knowledge generation [16].

4.1.1 Simulation-Based Search

A simulation-based approach is used in game playing that does not require any a priori domain knowledge.

4.1.1.1 Monte-Carlo Tree Search

Monte-Carlo (MC) methods root in statistical physics [20] used to obtain approximations to intractable integrals, in numerical algorithms being successful in various AI games. Monte-Carlo tree search (MCTS) is a best-first search technique that uses stochastic simulations, see Figure 4.3; it uses the true value of an action may be approximated using random simulations, and that value may be used to efficiently adjust the policy towards a best-first strategy [20]. The MCTS is one of the best-known examples of a simulation based search that makes use of MC simulation to evaluate the nodes of a search tree. The algorithm progressively builds a partial game tree, guided by the results of previous exploration of that tree; the tree is built to more accurately estimate the values of moves; the algorithm builds a tree according to the following mechanisms [33]: *selection*, *expansion*, *simulation* and *back-propagation*.

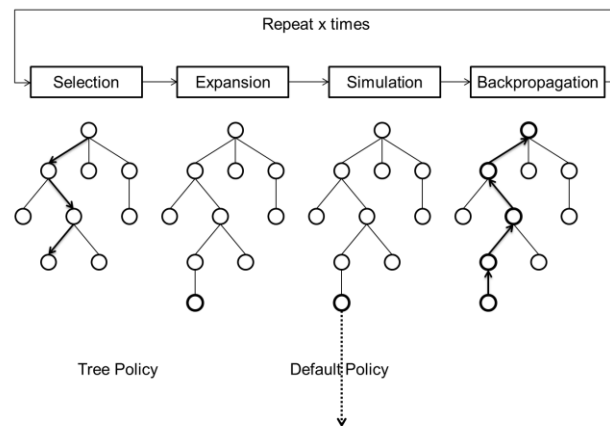


Figure 4.3 MCTS Approach, the process can be divided into: selection, expansion, simulation and back-propagation.

Gelly and *Silver* present two extensions to the MCTS algorithm [48]: The *Rapid Action Value Estimation* (RAVE) shares the value of actions across each sub-tree of the tree search, and the *heuristic MCTS* uses a function to initialize the value of the new positions in the tree search. Go programs based on MCTS now play at human-master level [47].

4.1.1.2 Temporal-Difference Learning

Temporal difference (TD) learning and reinforcement learning proved to be successful game playing techniques for other games, including Backgammon. The large state space of Go prevents these techniques from being effective when applied directly to the Go game.

TD *search* approach in computer Go [102] is another technique that has been used to automate Go players. TD learning is a prediction method, one of the most successful and broadly applied solutions to reinforcement learning problem. TD learning combines Monte Carlo Method and dynamic programming, because, it learns by sampling the environment according to a policy and approximates its current estimate based on previously estimates. Reinforcement learning is considered a slow procedure; it divides in two major problems: *learning* improves the agent policy with environment interactions and *planning* improves the agent policy without environment interactions. This approach combines TD learning with simulation-based search, essentially, with MCTS. It uses the mean outcome of simulated episodes of experience to evaluate each node in a search state, where each node corresponds a particular Go board configuration (state). TD search still lacks of efficiency, it requires many simulations per move to obtain good results but unlike MCTS, is seeking to integrate prior knowledge although MCTS has proved to be better.

4.1.2 Neural Network for Learning/Recognition

The ability of NNs to find hidden relationships from the input-output mapping of evaluated information makes this method powerful in areas where there exists a large number of patterns to analysis such as Go. The classic Back Propagation Neural Network (BPNN) method for NN training on pattern recognition uses a supervised learning to adjust the connection weights and is able to recognize complex pattern [101]. Typical topology of Multilayer BPNN is shown in Figure 4.4. Let $X = (x_1, \dots, x_i, \dots, x_n)^T$ be an input vector, $H = (h_1, \dots, h_i, \dots, h_m)^T$ be a hidden layer's output vector, $O = (o_1, \dots, o_i, \dots, o_z)^T$ an output layer's output vector, $T = (t_1, \dots, t_i, \dots, t_z)^T$, an expect output vector, $W = (w_1, \dots, w_i, \dots, w_m)^T$ a weight matrix form input layer to hidden layer, and $V = (v_1, \dots, v_i, \dots, v_z)^T$ a weight matrix from hidden layer to output layer.

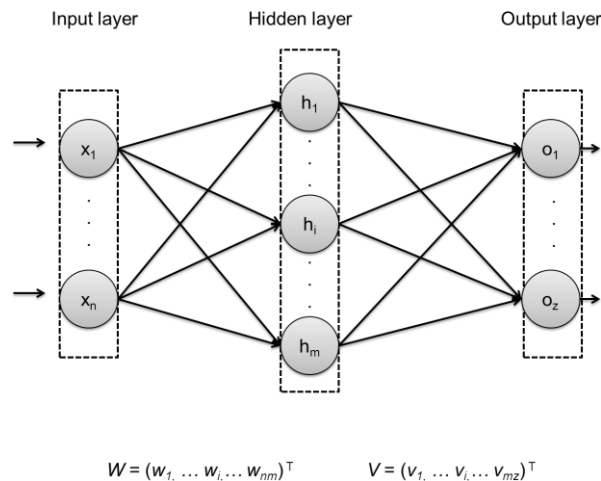


Figure 4.4 Multi-layer topology example.

SANE (Symbiotic, Adaptive Neuro-Evolution) [92] is used to evolve NNs capable of playing Go on small boards with no pre-programmed go knowledge. The major purpose is, combine Evolutionary Algorithms (EA) and NNs. The EA solves the problem of the credit assignment; this problem determines which moves played are good and deserve credit for a win and which are bad and deserve to be blamed for a loss. SANE evolves partial

solutions, *i.e.* neurons, represented as a collection of labeled weight links to input and output units. Evolving the label as well as the weight enables the genetic algorithm to evolve the structure of the network and the parameters. Those neurons, in turn, are grouped at random to form a network. One neuron can participate in more than one network. Hierarchical SANE evolves those groups of neurons as well as the neurons themselves.

The go-playing program *Honte* [36] uses NNs together with TD learning. One NN is trained to learn local shapes using supervised learning, additional, two more NNs are used by self-play using TD to estimate safety group and value the territorial potential of occupied points, *Honte* has not yet reached good level although the designing of NNs to imitate human concepts has been successful. When groups are not well defined because potential connections are still in question, a neural net trained with TD-learning evaluates their safety value. The groups are represented in terms of operational features such as the number of liberties, the number of eyes and the influence value of the liberties. Groups are considered safe if the stone remains on the board at the end of the game. Go AI program based on BP-Neural Network is analyzed in [60], in this approach presented a architecture of BP-NN, the authors claim that the use of NNs is for learning a model form training data, such as Go better than other techniques be very difficult .

A machine for playing Go –and related board games–, primarily focusing on the problem of learning a good evaluation function in a scalable way is developed in [109]. Scalability is essential at multiple levels, from the library of local tactical patterns, to the integration of patterns across the board, to the size of the board itself. System is automatically learning the propensity of local patterns from a library of games; propensity and other local tactical information are fed into recursive neural networks, derived from a probabilistic Bayesian network architecture. The recursive neural networks in turn integrate local information across the board in all four cardinal directions and produce local outputs that represent local territory ownership probabilities.

Evaluation of the current board position is critical in computer game engines. In [21] the combination of NNs, particle swarm optimization (PSO), and evolutionary algorithms (EAs) to train a board evaluator from zero knowledge is present. By enhancing the sur-

vivors of an EA with PSO, the hybrid algorithm successfully trains the high-dimensional NNs to provide an evaluation of the game board through self-play. Experimental results, on the benchmark game of capture Go, demonstrate that the hybrid algorithm can be more powerful than its individual parts, with the system playing against EA and PSO trained game engines.

Go gaming presents a broad set of shapes (diverse forms) that can be classified into desired patterns. Our approach focuses on the BPNN pattern recognition of Go tactics. This pattern recognition sets to strategic reasoning to decide some Go gaming actions. The pattern recognition process is essential to the learning and understanding of the game *moves* [84]. The better the understanding of game moves the better decisions can be made by playing. Moves can be learnt from patterns by humans then there should be also a way for computer programs.

4.1.3 Go Pattern Recognition

Shi-jim et al. [99] proposed a Go patterns matching algorithm to find game records that contains desired query patterns then an index structure for Go record database integrates methods of information retrieval and domains knowledge of Go. This approach mainly focuses on finding *edge* and *corner* patterns in to Go records and uses an index structure to increase the speed of pattern searching. To construct index structure on the game record database, the features of many significant query patterns is extracted and used as the index key. 400 featured patterns are used. Feature patterns are some patterns that appear frequently in Go games. The approach can solve the most difficult part of a Go game record information retrieval system. String matching for text in Go game records is easily implemented. The matching can be considered as text information retrieval.

Bayesian technique for supervised pattern-learning algorithm, based on the Bradley-Terry model, is proposed in [30]. The principle of Elo ratings, as applied to chess, is that each player gets numerical strength estimation, computed from the observation of past game results. From the ratings of players, it is possible to estimate a probability distribu-

tion over the outcome of future games. The generalized Bradley-Terry model is a powerful technique for pattern learning in the Go game, is simple and efficient, can combine several feature and produces a probability over legal moves, it can be used to incorporate domain knowledge into MCTS.

A systematic approach for knowledge representation of Go playing where the meaning of move is defined as a contextual pattern with respect to local contexts of surrounding of the move is introduced in [111]. The approach purpose is to acquire certain knowledge of patterns. Contextual patterns and pattern collocations have the following key advantages in comparison with other manual and ad hoc representations of GO-playing knowledge: Quality knowledge, efficient acquisition and effectiveness.

The recognition of strategies being performed by the player and the adversary is fundamental to propose the best offensive/defensive strategies. During the Go game match, different strategies and tactics are taking place, such as: *eyes*, *ladders*, *nets*, capture of adversary stones, among other. Artificial Neural Networks are powerful in pattern recognition especially Back-Propagation Neural Networks (BPNNs). BPNNs allow discovering and recognizing complex patterns and therefore strategies patter recognition is relevant to apply during a Go match.

During Go game match, an implicit complex analysis is being practiced by players to define the next move. Our aims are to incorporate mechanisms that help to identify the best options (strategies) to perform in certain Go match stages. After a number of moves, the BPNNs should recognize what the player and adversary are doing for identifying the possible offensive/defensive strategies. The strategic analysis is used by both players.

4.1.4 Evolutionary Algorithms

The *GoTools* program solves life and death problems in the Go game. *Pratola* and *Wolf* [90] introduced Genetic Algorithm (GA) to optimize heuristic weight used by *GoTools* tree-search. The set of heuristic weights is composed of subgroups, each optimized with a suitable fitness function. In order to optimize the heuristic weights, and hence the use-

fulness of their heuristic rules, a GA was implemented to search for the best set of heuristic weights. As they have many heuristic rules available, they also have many heuristic weights to consider. A set of these heuristic weights is referred to as a *chromosome*. Thus, each chromosome is a candidate solution, containing a set of heuristic weights (whose individual values are referred to as *allele's*) to be optimized.

Shiba et al. [100] proposed a system which automatically carries out Go record from images which pictured the Go game situation, from the Go stone position choice on the 19 x 19 cells of the Go board. A genetic algorithm for Go board contour detection in the real image is used, that show the elite saving strategy mode for combining a global and local search.

To find moves in Go with representation of partial strategies as a multithreaded, steady-state co-evolution algorithm without domain-specific knowledge can discover the correct move [40]; valuable performing distributed tree search, quickly eliminating unpromising moves and dividing exploratory computation across multiple search paths are claimed. This approach rather than evolve a general function for playing Go from any board position, it evolve partial strategies for the current position.

Lei Yu et al. [67] calculate the winning probability which model parameter is optimized by genetic algorithm. The winning probability model presented by this paper has a high precision, and it can be applied in the middle game module and endgame module, helping the program to determine the best move. This approach mainly focuses on the leading points and Go situation, combined with genetic algorithm and optimizing the model parameter.

Solving multi-objective optimization problems using evolutionary algorithms has been gaining a lot of interest recently. Go is a hard and complex board game. Using EAs, a computer may learn to play the game of Go by playing the games repeatedly and gaining the experience from these repeated plays. In [62], artificial neural networks (NNs) are evolved with the Pareto Archived Evolution Strategies (PAES) for the computer player to automatically learn and optimally play the small board Go game. ANNs will be automatically evolved with the least amount of complexity (number of hidden units) to optimally play the Go game. The complexity of NN is of particular importance since it will

influence the generalization capability of the evolved network. Hence, there are two conflicting objectives in this study; first is maximizing the Go game fitness score and the second is reducing the complexity in the NN. Several comparative empirical experiments were conducted that showed that the multi-objective optimization with two distinct and conflicting fitness functions outperformed the single-objective optimization which only optimized the first objective with no selection pressure selection on the second objective.

4.1.5 Planning Techniques

In [107] proposed an adversarial planning approach called goal-driven. In this approach is described how adversarial Hierarchical Task Network (HTN) planning can provide a framework for goal-directed game. Each player attempts to satisfy its own goals while refuting those of its opponents. If one player achieves its goal, the opponent backtracks and tries a different decomposition, so each player keeps an open agenda of goals which represents its current plan of action. To solve a problem in a domain, each player is given an abstract goal (or set of goals) to achieve. The players then attempt to find sequences of moves which satisfy their goals. Once one of the players has achieved all of its goals it knows that it must have satisfied its top level goals. The opposing player is made aware of this fact and, since in general a good outcome for one player is a bad outcome for the other, both agents are allowed to force backtracking. Players are allowed to backtrack to any of their previous goal or expansion choices but only to their own decisions.

4.2 Mathematical Modeling

This section is devoted to characterize the Go gaming algorithmic perspective, using process diagrams, and finite state machines that support our Go game approach.

Mathematical descriptions of Go game concepts follow,

- Board: $T = \{(x, y) | 1 \leq x, y \leq n\}$, $n = 19$ is the official size.

- Let $\pi \in T, \forall \pi, \rho(\pi) \in \{e, b, w\}$, $\rho(\pi)$ denotes the content of a coordinate; in the beginning all $\rho(\pi) = e$ an empty position in T , and b, w denote black and white, $\Lambda = \{b, w\}$.
- $Ps = \{\pi \mid \pi \in T, \rho(\pi) \in \Lambda\} \neq \emptyset$, is the set of occupied positions.
- $Mh(\pi_0, \pi_1) = |\pi_0.x - \pi_1.x| + |\pi_0.y - \pi_1.y|$ denotes the Manhattan distance between $\pi_0, \pi_1 \in T$.
- Stone: $st = (\pi_1, \pi_2 \dots \pi_n), n \in \mathbb{N}, \rho(\pi_1) = \dots = \rho(\pi_n)$, and $Mh(\pi_i, \pi_{i+1}) = 1$. A single stone is $n = 1$, otherwise is a sequence of single stones each another adjacent.
- Set of stones: $St = \{st \subseteq Ps, \mid st \text{ is a stone}\}$
- Set of liberties of $st \in St$: $L(st) = \{\pi \in T \mid \rho(\pi) \in T - Ps, \text{ and } \exists \pi_0 \text{ member of } st \text{ with } Mh(\pi_0, \pi) = 1\}$.
- *atari*: $st \in St$ is in *atari* if and only if (iff) $|L(st)| = 1$
- *eye*: $|\bigcap_{i=1}^n L(st_i)| = 1, n = 2,3,4$.
- *Ladder*: $ld \in St$ is a *ladder* on st iff st is the other color to ld and is in *atari* by ld .
- *Net*: $nt \in St$ is a *net* on st iff $\forall b \in L(st), \exists \hat{st}$ in nt and π in \hat{st} such that $Mh(b, \pi) = 1$; that is, any liberty of st is adjacent to an adversary's stone.
- $P = \{p_1, p_2\}$, p_1 and p_2 the black- and white-player.
- $A = \{a_1, a_2, \dots a_m\}$ is the set of actions.
- $\varphi: S \times A \rightarrow S$ is the transition function or relation.
- S is the state space.

We characterize the algorithmic description of the Go gaming using all the previous mathematical definitions.

4.3 Algorithmic Modeling

In Game Theory, the formal modeling of gaming accounts for the interaction between the players' actions by obeying the game rules. Match gaming is algorithmically implemented according to the interaction of the strategies E^i each player applies in attempting to achieve maximum gain. We give a formal definition of strategy that will be used later to define basic tactics and some basic strategies.

A strategy is formally defined as $\ddot{e} = (a_1, a_2, \dots, a_n)$ so $\ddot{e} \in A^n$ is a sequence of planned actions. For $p_i \in P$ let:

- $E^i = \{e_1, e_2, \dots, e_n\}$ the set of strategies.
- $E = E^{p_1} \times E^{p_2}$ is the strategy space.
- $\tau^i = \{t_1, t_2, \dots, t_o\} \subseteq E^i$ the set of tactics (simple strategies).

A finite state machine (FSM) for Go strategies algorithmic setting in equation (4.1) is defined as follows, see Figure 4.5:

$$F_{\ddot{E}} = (\bar{A}, \hat{S}, s, \hat{\varphi}, H) \quad (4.1)$$

$\bar{A} \subseteq A$ is a set of symbols that denote basic actions, $\hat{S} = \{s, s_0, s_1, \dots, s_l, h\}$ is the set of states, $\hat{\varphi}: \hat{S} \times \bar{A} \rightarrow \hat{S}$ is the transition function. $H = \{h\}$ is the set of final states, $s \in \hat{S}$ is the initial state.

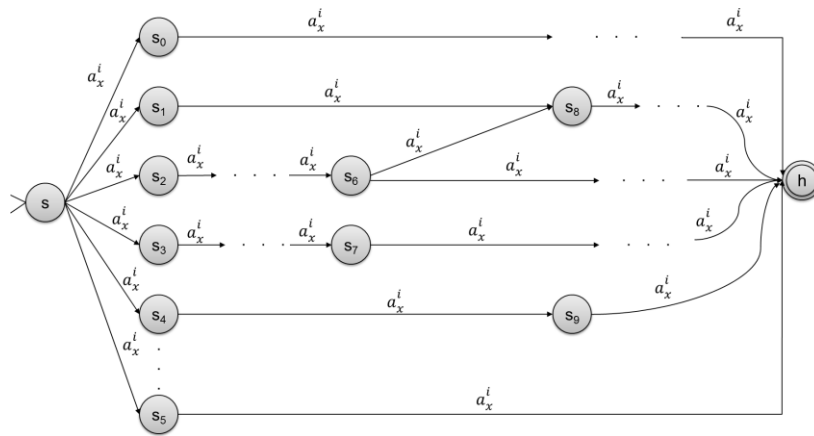


Figure 4.5 FSM for modeling the player' strategies from basic actions a_x^i .

This FSM computes any strategy for a $p_i \in P$, by passing through states until the final (halt) state h ; at each level of automata, actions a_x^i are different because it is a deterministic FSM, so given a state and action, there exist one and only one transition to a next state in equation (4.2).

$$e_x^i = (a_1^i, a_2^i, \dots, a_p^i) \quad (4.2)$$

A tactic $\check{t} \in \tau^i$ is defined by $\check{t} = (a_x)$. A FSM for Go tactics (4.3) follows see Figure 4.6:

$$F_{\check{T}} = (\bar{A}, \hat{S}, s, \hat{\phi}, H) \quad (4.3)$$

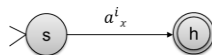


Figure 4.6 The FSM modeling the a_x^i player's tactics.

FSMs that model eyes, ladders and nets is in equation (4.4).

$$F = (\bar{A}, \hat{S}, s, \hat{\phi}, H) \quad (4.4)$$

See Figure 4.7 for the FSM for modeling eyes. Chains halting at h_0 and holding eye conditions are edge eyes; and those halting at h_1 and h_2 and holding eye conditions represent *lateral* and *normal* eyes respectively. Here:

$$\hat{S} = \{s, s_0, s_1, s_2, h_0, h_1, h_2\} \text{ and } H = \{h_0, h_1, h_2\}.$$

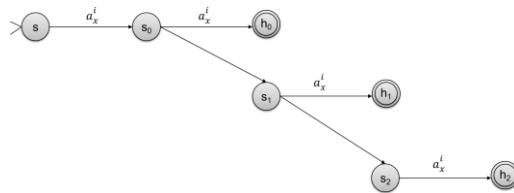


Figure 4.7 FSM for eyes creation.

In the FSM for ladders creation, $\hat{S} = \{s, s_0, \dots, s_z, h\}$, and $H = \{h\}$. Any string of symbols halting at h and holding the ladder conditions, i.e., forcing an adversary's stone in atari, result in a ladder, see Figure 4.8:



Figure 4.8 FSM for ladder creation.

In a FSM for nets creation, any string of symbols halting at h and surrounding adversary's stones results in a net, see Figure 4.9:

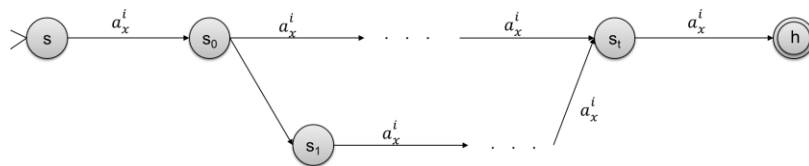


Figure 4.9 FSM for net creation.

The diagram flow that indicates the gaming states/moves is in Figure 4.10.

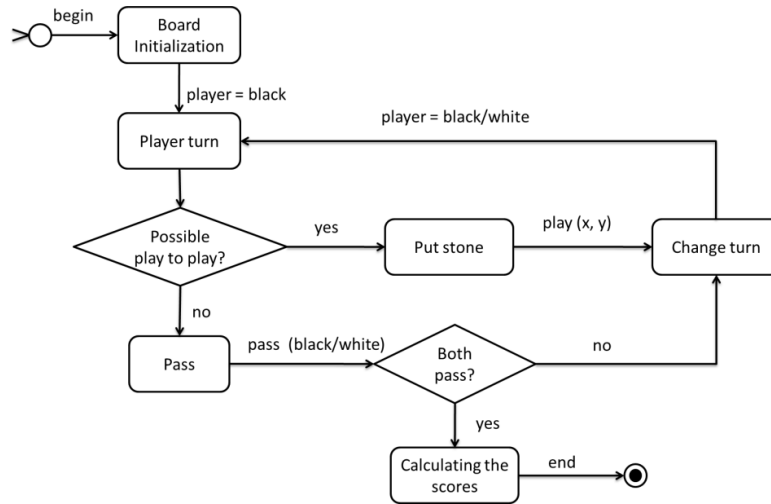


Figure 4.10 Go game process flow.

In this proposal, it uses MCTS simulations, all the actions have the same probability, so uniform distribution is used (4.5) and (4.6):

$$p(a_i) = \frac{1}{m} \quad (4.5)$$

$$\sum_{i=1}^m p(a_i) = 1. \quad (4.6)$$

4.4 Tactics Recognition and Strategies Building

Strategic reasoning is used to decide, from the learned Go actions, a convenient move to do in the current state of the match, so the next node in the match-game decision tree, as a result from recognizing the adversary's Go tactics being deployed so far. After an empirical analysis of Go matches by stages, we concluded that:

- In the early and middle Go match stages to apply pattern recognition and strategic reasoning is an efficient gaming option, since is the way for using *a priori* known tactics and strategies. Actually, it serves as the basis to deploy a match gaming.
- In the late Go match stage free positions on the board are too restrictive, so the deployment of *a priori* strategies is difficult; under this circumstance the gaming way is by doing a MCTS evaluation to play any of the board free positions.

Thus, during the early and middle a Go match stages, our automated Go player uses pattern recognition to identify the adversary's (sequence of) movements, so identify his/her strategies followed so far; a strategic answer should be reasoned to decide the next movement using a priori knowledge from the players' experience. Note that, in a Go match early stage the set of board free positions is high and any movement by MCTS tends to be random, so, due to huge size of search space the processing is time spending. At this match early stage MCTS is not an efficient technique but the opposite. However, in the late Go match stage a MCTS move becomes an efficient option: the size of search space has become small and, on the other hand, the automated pattern recognition is too hard to do over the remaining board free positions.

4.4.1 Tactics Pattern Recognition

For the process of pattern recognition analysis, the Go board is segmented into a *window view size* of 3×3 in order to identify eyes patterns. A *window view size* of 5×5 is used for identifying ladders and nets patterns, and as a result of the neighboring combination of these windows, bigger ladders and nets can be recognized. The NN layer of the input receives a set of board occupied points; 9 for eyes and 25 for a ladder or a net. During the training stage, the training patterns include *don't care* symbols to represent those points that can be replaced regardless of their value, see examples in Figure 4.11. It is valuable to include *don't care* patterns in Go tactics recognition because of the non-deterministic nature of Go gaming.

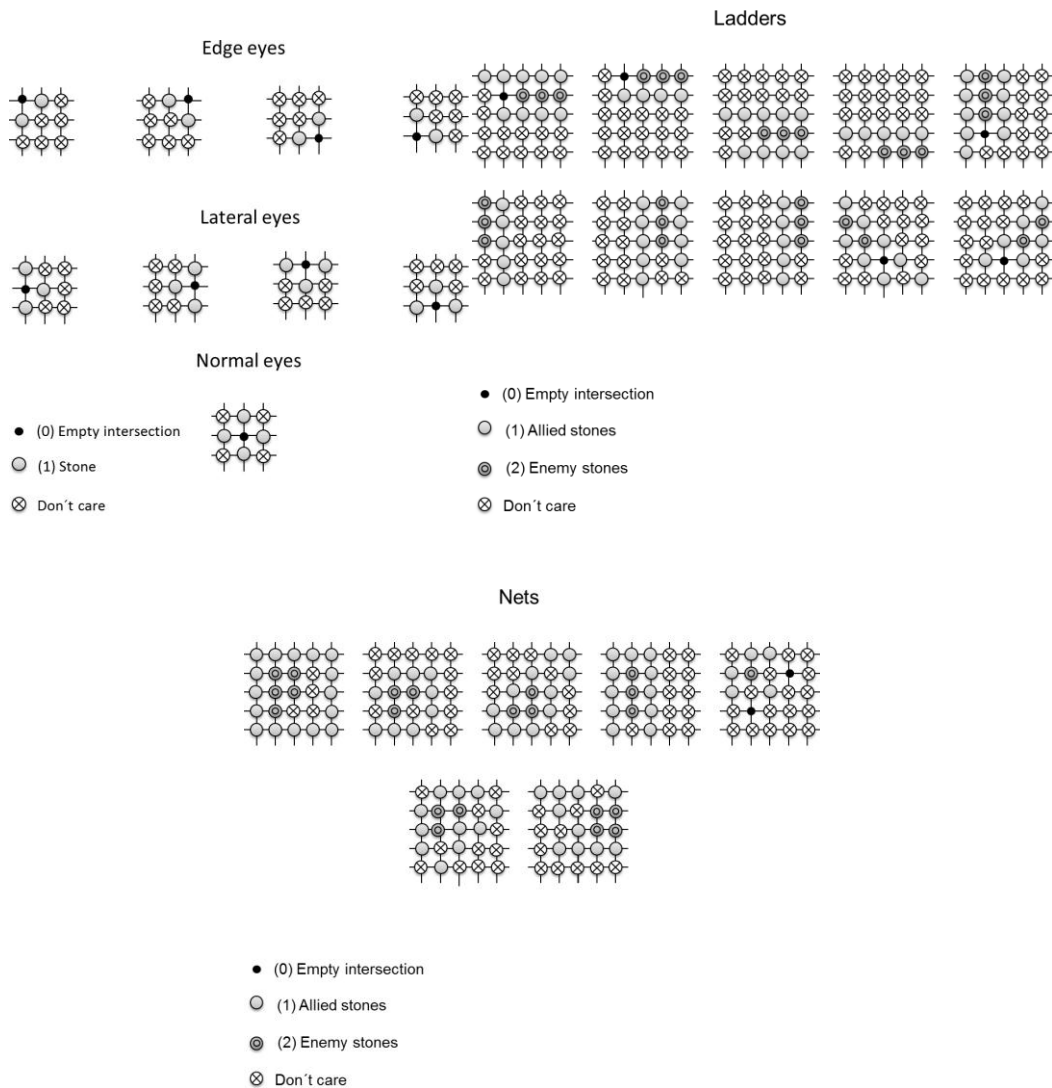


Figure 4.11 Illustrations of eyes, ladders and nets patterns.

To start the process of Go tactics pattern recognition, the positions from the 3×3 and 5×5 windows view size are encoded into a vector that feeds the network. When using pattern recognition to identify Go tactics in a match, the main difficulty concerns verifying that a shape really corresponds to an eye, ladder or net. The NN should check the conditions to authenticate whether the detected shape is a true Go tactic.

For eye pattern recognition, the NN tries to find similarities with the given input to any of the shapes described in Figure 4.11: edge, lateral or normal eyes. If high similarities exist then the conditions of eye must be checked, i.e., there must be an empty point of

space surrounded by friendly stones such that no adversary's stone may be set upon it. These conditions are verified outside the NN using verifier conditions. As in the case of eye patterns, the same procedure is applied for ladder and net pattern recognition, but with the proper ladder and net conditions.

4.4.2 Building of Strategies from Pattern Recognition

Once the Go tactics patterns such as eyes, ladders or nets, are recognized, as well as the Go gaming strategies of invasion, reduction, connection or capture, offensive/defensive strategies can be employed (see Figure 4.12). Hence, based on the tactics pattern recognition, deployment heuristics for suitable defensive/offensive Go *a-priori-knowledge* strategies are available to be applied during the initial and middle steps of an automated Go match. Strategies of reduction and invasion as well as defensive strategies, address saving stones in atari or are devoted to augment the liberties of ally stones. Strategies can be constructed following the next statements, as illustrated in Figure 4.13.

Defensive strategies:

- Save a stone in atari by close placement of an ally stone that eventually connects and saves it, or increasing liberties.
- For preventing a stone falling within risk of be captured by the adversary's next moves.

Offensive strategies:

- Interrupt the formation of adversary's eye by placing a new stone.
- Reduce liberties to adversary's stones, eventually placing it in atari.
- Play close to own stones, sets of stone(s) or close to stone(s) with two or more eyes to ensure high possibilities of making connections and spreading of stones.
- Capture adversary's stones by placing adversary's stones in or close to atari.

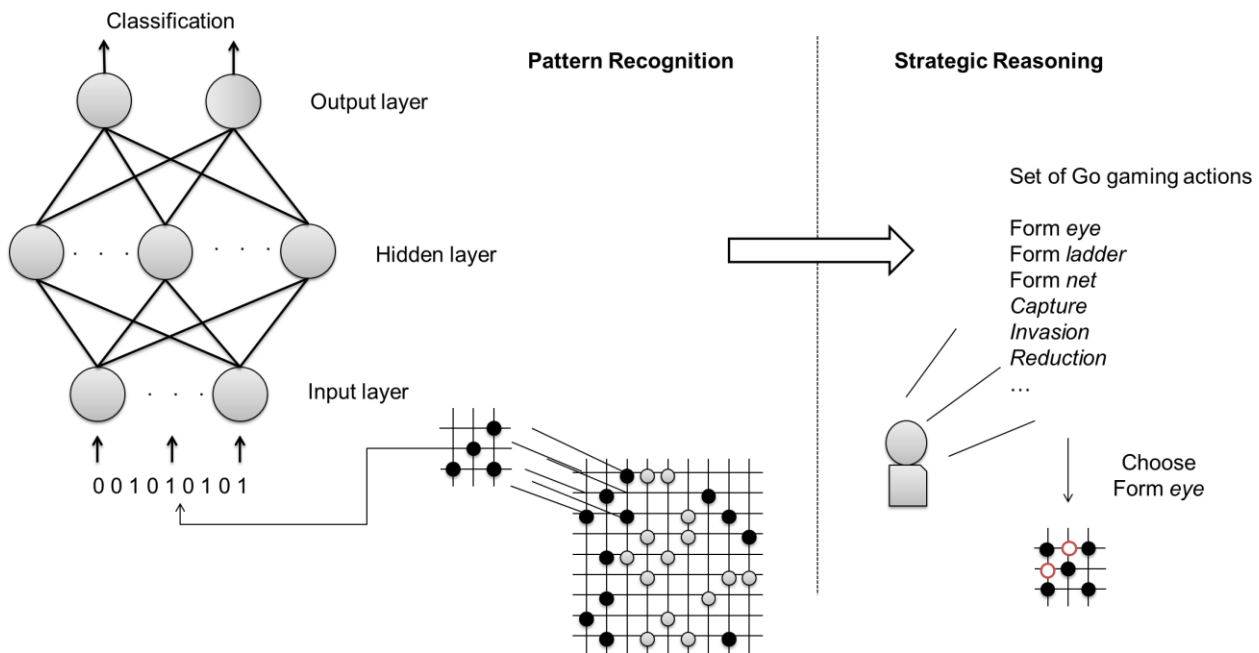


Figure 4.12 Example pattern recognition of a possible eye.

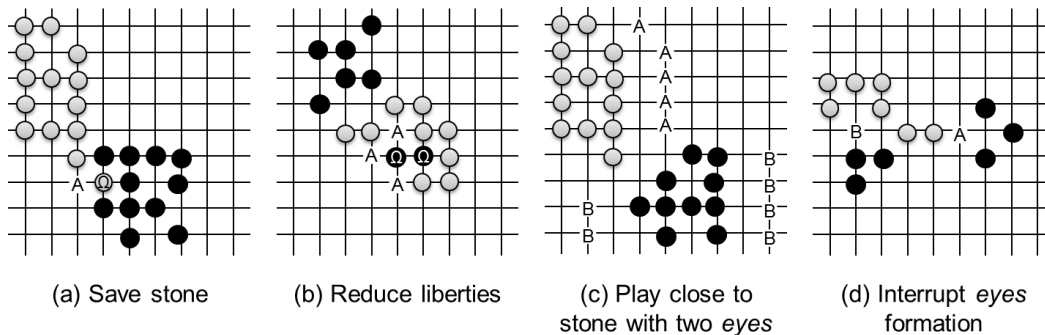


Figure 4.13 Go gaming strategies: (a) Ω is in atari, but playing in point A makes a connection for saving Ω ; (b) playing in points A reduces Ω liberties; (c) white/black playing to points A/B increases dominance area; (d) white/black playing to points A/B interrupts the formation of adversary's eye.

In the latter stages of a Go match, an MCTS-based move becomes a suitable option. This is because the size of the search space has become small and the automated pattern recognition is too difficult to perform over the complex patterns on the board with the few free board positions. Actually, in the latter stages of a Go match, the deployment of *a-priori*-known strategies is hard because the free board points are too restrictive, and

few board spaces make it difficult to deploy strategies. Under this circumstance, the gaming method is to perform an MCTS evaluation to play any of the free board positions. Hybrid approaches with *Go a-priori-knowledge*-based strategies are easy to deploy in the initial and middle stages of the game when few board points are occupied. By the end stages, the use of MCTS is better suited to choosing a move on the empty board positions. A description of the hybrid Go players that prove our claim is given in the following.

4.5 Strategic Players

Our gaming simulator compromises a set of automated Go players, using either random, or pattern recognition, or strategic reasoning, or MC-Rave methods. In addition, it has a graphic interface and uses Go Text Protocol for communication with other automated players in KGS [26] Go servers (see Figure 4.14).

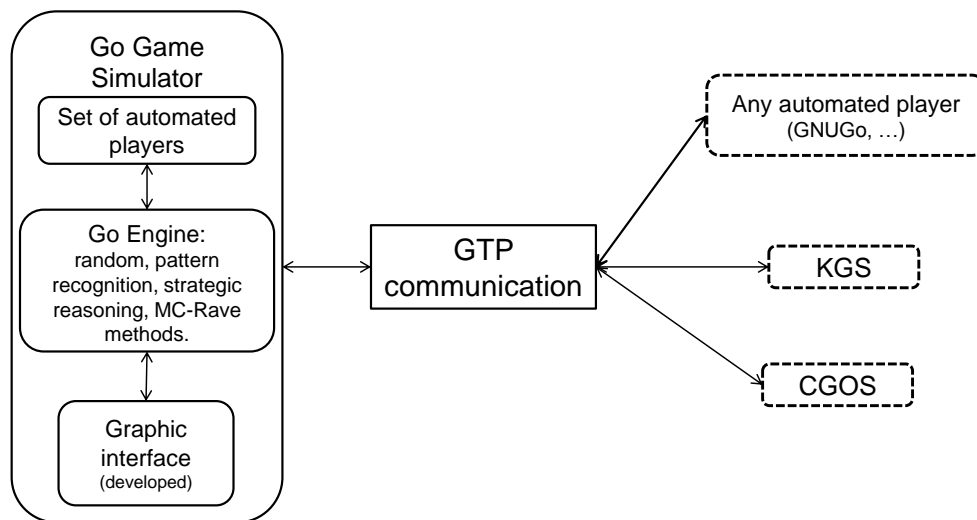


Figure 4.14 Components of Go gaming simulator.

4.5.1 Hybrid SP_1 Player with MCTS

Our next strategic player SP_1 combines the pattern recognition with the MCTS Rave approach [48], see Figure 4.15.

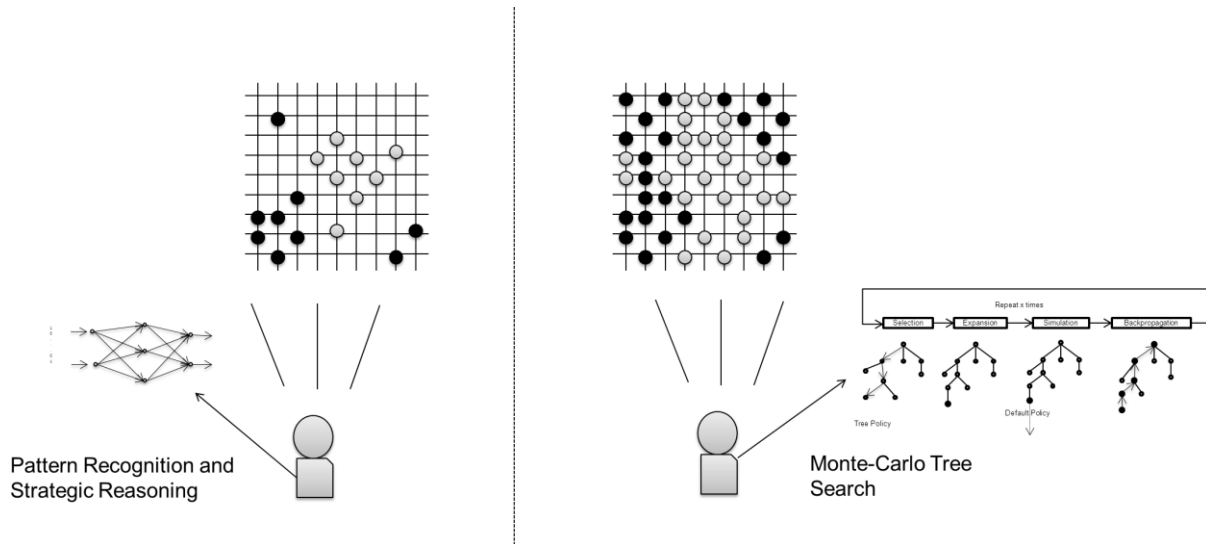


Figure 4.15 Hybrid Go gaming: pattern recognition of tactics and strategies by the early and middle match stages, and MCTS in the latter stages.

This SP_1 hybrid use experience-known Go tactics patterns in the early and middle stages of a match, and MCTS in the late stage.

- To start the process of Go tactics pattern recognition, the positions from 3 x 3 and 5 x 5 *windows view size* are encoded into a vector that feeds the network.
- Once a certain pattern is recognized, a set of Go gaming actions is proposed as offensive/defensive strategies, see Figure 4.10. Go gaming actions like do eyes, ladders, nets, invasion, reduction, connection, capture, are performed as offensive/defensive strategies.

Using pattern recognition to identify Go tactics in a match a main difficulty concerns verify a shape really corresponds to an eye, ladder or net. The NN should check conditions to authenticate if the shape is a true Go tactic.

4.5.2 Hybrid SP_2 Player with *GNUGo*

GNUGo is a classic and powerful automated open source Go player, the first with huge impact in Go automation [14], so nowadays, comparison versus *GNUGo* is obligated; *GNUGo* uses pattern matching algorithms to analyze the stones patterns on the board then proposing next moves [44]. In addition:

- *GNUGo* rates the shape formed by a move, and the value of local moves at specific territory,
- *GNUGo* calculates the max/min value obtained by a move using pattern matching.

Because the *GNUGo* Go player strength is in the criteria used to evaluate a move, our next proposed hybrid player SP_2 is based on *GNUGo*. SP_2 uses *GNUGo* criteria to play during the first 10 moves and then shifts to pattern recognition for the subsequent 10 moves, on average, and so on. The reason for doing 10 moves each is because the match stages are determined by the number of moves played by each player. Statistically, for a 9×9 board size, the number of moves is around 40 per player and therefore, on average, each early, middle or late stage in a match is scoped by 13 moves per player. Similar calculus works for a 19×19 size board.

4.6 Go Gaming Experiments and Comparisons

The average recognition performance obtained by different numbers of hidden neurons is shown in Figure 4.16. In order to surpass the major difficulty for Go tactics patterns recognition, given the wide variety of shapes in a Go match; we fix on five for the number of neurons in the hidden layer of the multi-layer NN used. This way, efficient pattern recognition and learning is our best performance. Furthermore, the training time is short enough to avoid over-learning, which produces noise and/or redundancy. The test description to recognize eyes, ladders and nets on the Go board follows.

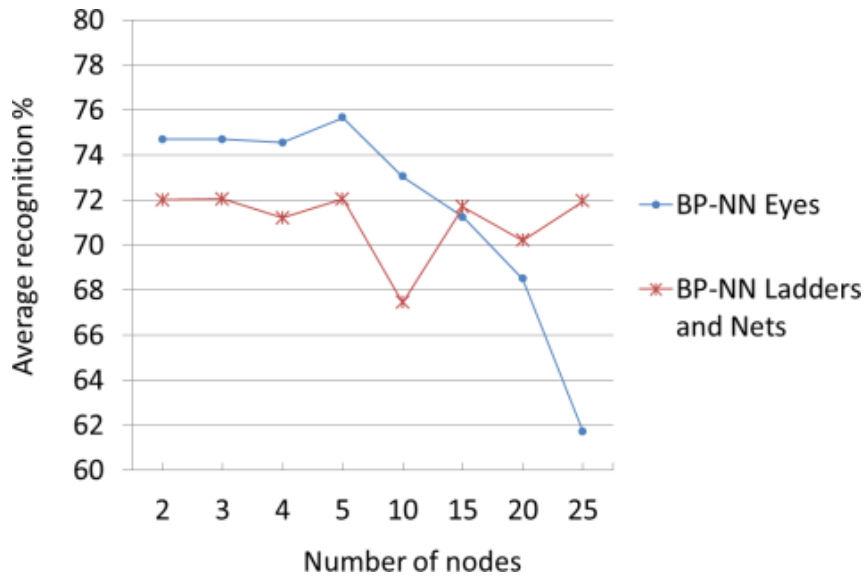


Figure 4.16 Tradeoff: number of neurons in the hidden layer versus the recognition percentage.

Test description to recognize eyes, ladders and nets on the Go board follows.

Eyes patterns. 900 tests divided into groups of 150 were performed. 400 eyes positive examples and 100 not-eyes negative examples were used to test the NN. The mean, minimum and maximum accuracy obtained for each group from the eyes/no-eyes examples and the mean accuracy of the NN are reported. The average accuracy is above 70%, as shown in Table 4.2.

Table 4.2 Results of NN eyes.

| Number of tests | Eyes examples | | | No-eyes examples | | Total classification Eyes + No-eyes | |
|-----------------|----------------------------------|------|------|------------------|------|--|-------|
| | (Correct classification cases %) | | | | | | |
| | Mean | Min. | Max. | Mean | Min. | Max | Mean |
| 1 -150 | 83.68 | 80 | 90 | 67.58 | 58 | 74 | 75.77 |
| 151-300 | 83.24 | 80 | 88 | 68.04 | 60 | 74 | 75.64 |
| 301-450 | 83.58 | 80 | 91 | 67.72 | 56 | 74 | 75.65 |
| 451-600 | 83.78 | 80 | 91 | 67.45 | 60 | 74 | 75.62 |
| 601-750 | 83.62 | 80 | 90 | 67.69 | 60 | 74 | 75.65 |
| 751-900 | 83.32 | 81 | 88 | 67.96 | 60 | 74 | 75.64 |

Ladders and Nets patterns. A similar analysis was done for ladders and nets by applying 900 tests. Each test uses 100 ladders-nets positive examples and 100 no-ladders-nets negative examples. 75% accuracy is obtained, see Table 4.3.

Table 4.3 Results of NN Ladders and nets.

| Number of tests | Ladders-nets examples | | | No-ladders-nets examples | | | Total classification Ladders-net + No-ladders-nets |
|-----------------|----------------------------------|------|------|--------------------------|------|------|--|
| | (Correct classification cases %) | | | | | | |
| | Mean | Min. | Max. | Mean | Min. | Max. | Mean |
| 1 -150 | 86.96 | 60 | 100 | 56.06 | 21 | 75 | 71.51 |
| 151-300 | 87.19 | 61 | 100 | 56.16 | 30 | 75 | 71.67 |
| 301-450 | 86.7 | 48 | 100 | 56.89 | 26 | 90 | 71.79 |
| 451-600 | 85.88 | 29 | 100 | 57.58 | 25 | 100 | 71.73 |
| 601-750 | 86.34 | 43 | 100 | 56.83 | 25 | 90 | 71.58 |
| 751-900 | 87.78 | 63 | 100 | 55.09 | 22 | 80 | 71.44 |

The experimental results prove certain improving on the NNs effectiveness to recognize complex patterns of Go gaming. Actually, the pattern recognition is through a wide variety of shapes, sometimes too complex and not obvious of true Go tactics. Results in Tables 4.1 and 4.2 show the NNs accuracy. The around 70% of efficiency recognition is a good result due to the complexity of these kinds of patterns. The huge amount of forms that occur by a Go gaming match makes the tactics patterns recognition a hard task. Even that the recognized patterns correspond to tactics that are significant for the proposed Go strategic analysis. The [99] approach tries to find out Go patterns in game records like *edge* and *corner* patterns but few of them represent proper Go tactics patterns.

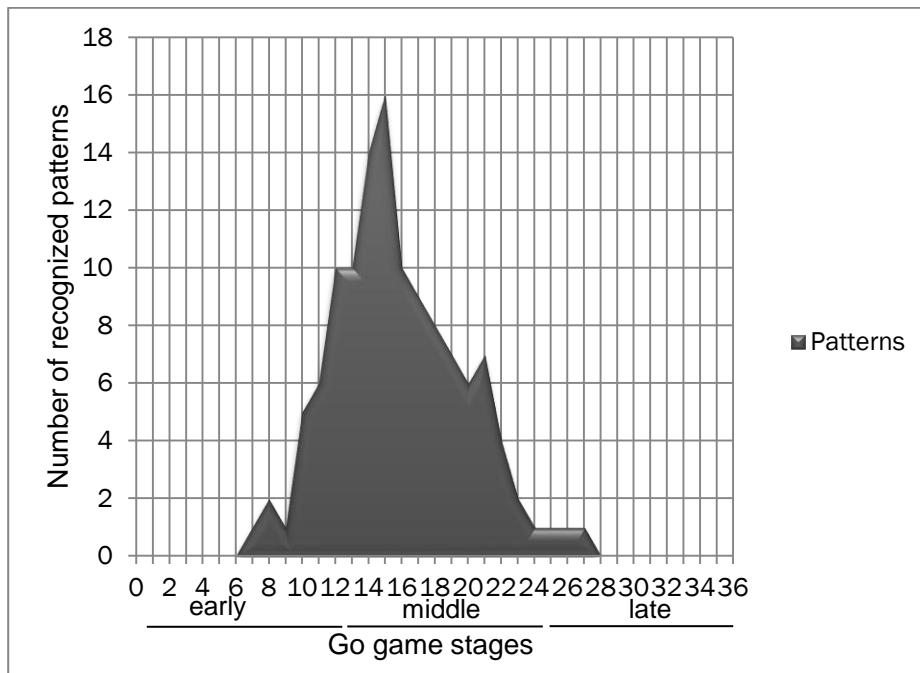


Figure 4.17 Number of patterns recognized through stages in a Go match.

Figure 4.17 shows that the highest number of recognized patterns occurs in the middle stage of the match. In the early and middle stages, based on the Go gaming *a-priori-knowledge* of these states, the pattern recognition and strategic reasoning work; thus, a better strategic analysis of offensive/defensive Go actions is available. However, when we lack information or when the free board positions are too restrictive, corresponding to typical circumstances of the latter stages of the game, the use of MCTS on the set of free positions performs best for choosing the best play. Note, the numbers on the x-axis represent Go match turns for both players.

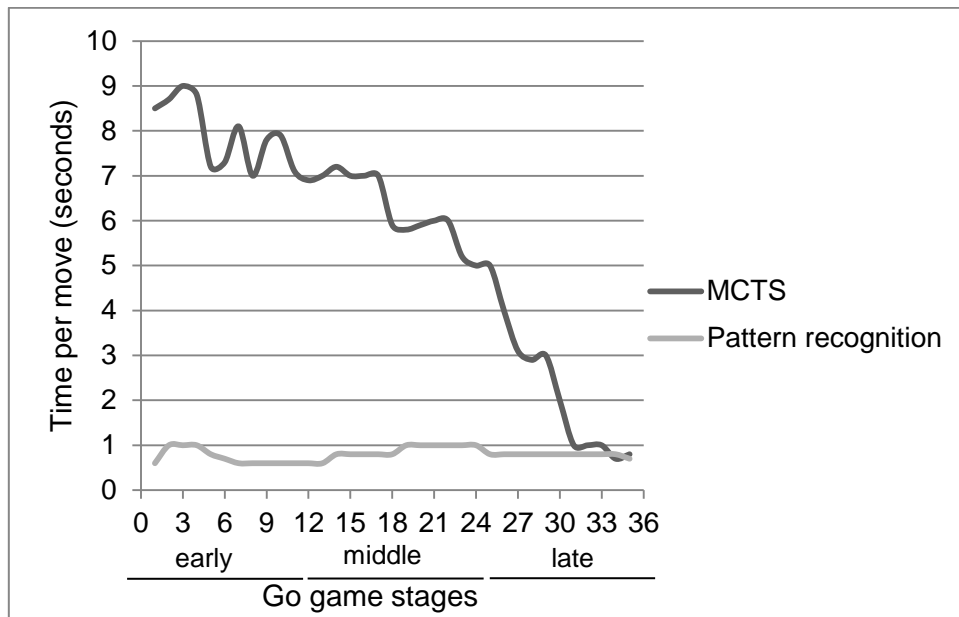


Figure 4.18 Elapsed time per move throughout stages of a Go match.

Figure 4.18 shows the elapsed time per move per player throughout the stages of a Go match. Time required per move using pattern recognition is too low and almost constant during the first and middle stages, but increases significantly in the latter stages because of the difficulty in recognizing any patterns on the board during this stage. In contrast, by using MCTS, the time spent deciding of a move is too high in the early stage, but reduces in the middle stage and is truly short in the late stage. The reason for this is that the size of search space the algorithm works in the early stage is huge; thus, applying MCTS is expensive and wastes a lot of time. In the late stage of a Go match, the search space size is small and so applying MCTS is faster.

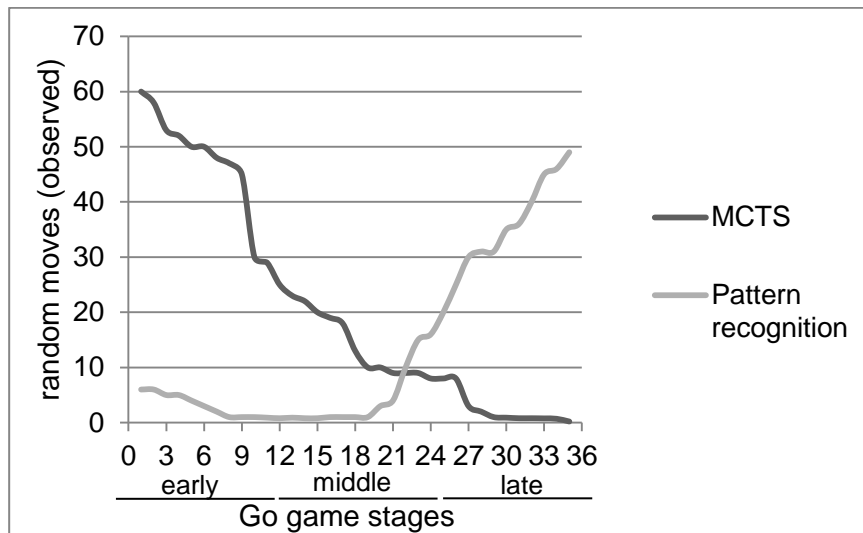


Figure 4.19 Percentage of random moves throughout stages of a Go match.

Figure 4.19 shows the percentage of observed random moves in 100 simulations. In the late stage, MCTS movements are suitable because each position can be evaluated quickly and therefore, it is the best method. In the early and middle stages, MCTS is too time consuming, but the random pattern recognition supporting *a-priori*-known strategic Go movements are easy to apply.

From the experiments described, we can conclude that in the early and middle stages of a Go match, it is best to apply *a-priori-knowledge* for the pattern recognition of tactics and strategies; thus, efficient gaming is achieved this way during these match steps. Conversely, because in the early stages of a Go match the number of free board points is large, any movement by MCTS tends to be random, and the huge size of the search space means that the processing time is high. Hence, during the early stage of a match, the MCTS technique is inefficient and the computer resources are wasted. Henceforth, we propose the systematic use of hybrid Go automated players for achieving efficient performance during matches.

4.7 Players' Performance Analysis

The performance comparison among the different automated Go players and the analytical comparison of approaches that use NNs for Go automation follow. The compared Go automated gamers are **SP_1** (uses pattern recognition, strategic reasoning and MC-Rave [48]), and the **Strategic player** that uses pattern recognition and strategic reasoning, and **MC-Rave player** that uses the method in [48].

4.7.1 SP_1 Performance Comparison

1000 tests on board size of 9 x 9 of the following combination were performed to make a comparison among them:

- (a). SP_1 vs. SP_1 ,
- (b). SP_1 vs. Strategic player,
- (c). Strategic player vs. SP_1 ,
- (d). SP_1 vs. MC-Rave,
- (e). MC-Rave vs. MC-Rave.

In Figure 4.18, the performance of the automated players from 1000 simulations is shown. Black SP_1 vs. white SP_1 is 49.3%/50.7% of wins rate (see Figure 4.20 (a)). Black SP_1 vs. white Strategic player is 53%/47% of wins rate (see Figure 4.20 (b)). Black Strategic player vs. white SP_1 is 44.7%/55.3% of wins rate (see Figure 4.20 (c)). Black SP_1 vs. white MC-Rave player is 52%/48% of wins rate (see Figure 4.20 (d)). Black MC-Rave player vs. white MC-Rave player is 48.9%/51.1% of wins rate (see Figure 4.20 (e)).

As shown in the results, SP_1 overcomes the other automated Go player's performances by applying different techniques in the early, middle, and late stages of the Go match.

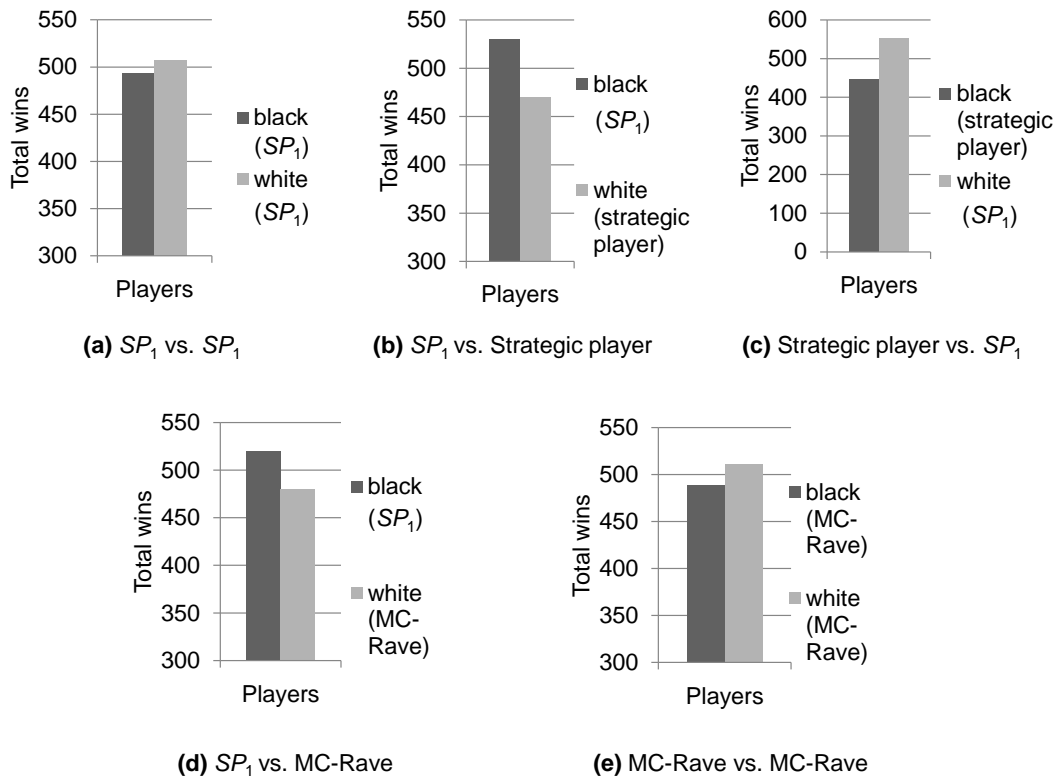


Figure 4.20 First set of performance results of automated Go players.

4.7.2 SP_2 Performance Comparison

1000 tests on board size of 9 x 9 using the following players were performed: **SP_2 (HybridGNUGo) player** uses pattern recognition, strategic reasoning and GNUGo, **GNUGo player**, **Strategic player** and **Hybrid player (SP_1)** uses pattern recognition, strategic reasoning along with MCTS. The performance comparisons were done as follows:

- (a). SP_2 vs. SP_1 ,
- (b). SP_2 vs. GNUGo,
- (c). SP_2 vs. Strategic player,
- (d). GNUGo vs. SP_1 ,
- (e). GNUGo vs. Strategic player.

The automated players' performances from 1000 simulations are shown in Figure 4.19. Black SP_2 player vs. white SP_1 is 55.7%/44.3% of wins rate (see Figure 4.21 (a)). Black SP_2 player vs. white GNUGo player is 51.1%/49.9% of wins rate (see Figure 4.21 (b)). Black SP_2 player vs. white Strategic player is 61.27%/38.8% of wins rate (see Figure 4.21 (c)). Black GNUGo player vs. white SP_1 is 54.5%/45.5% of wins rate (see Figure 4.21 (d)). Black GNUGo player vs. white Strategic player is 58%/42% of wins rate (see Figure 4.21 (e)).

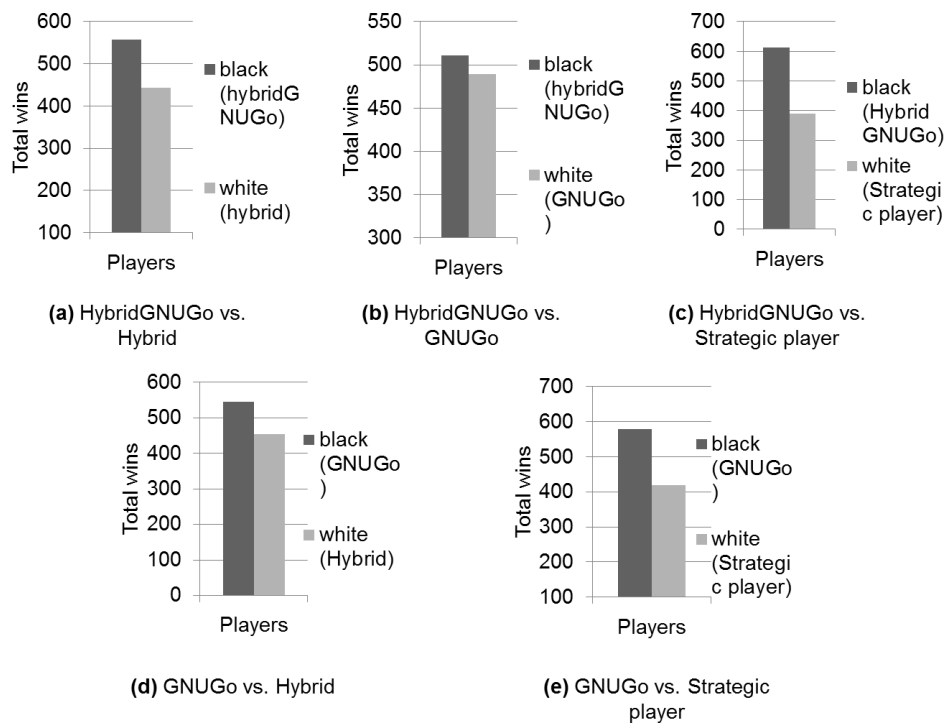


Figure 4.21 Second set of performance results of automated Go players.

The results show that SP_2 overcomes the other automated Go player's performances, even the GNUGo performance, by applying different techniques in the early, middle, and late stages of a Go match.

The Elo rating estimates the abilities level of Go player [30]. We measure the Elo rating of our automated Go players on Computer Go Server (CGOS) on 9 x 9 board size, with different *on-line* adversaries. The Elo ratings reported, see Table 4.4, is from average

values. Strategic players performance was also compared versus human players with low Go level, although this analysis is not reported because is not representative enough, particularly, all matches were in favor of our automated players.

Table 4.4 Elo Rating comparison.

| Automated player | Elo rating on CGOS |
|-------------------------|---------------------------|
| <i>Strategic Player</i> | 810 |
| SP_1 | 1120 |
| SP_2 | 1650 |

4.7.3 NN-Based Approaches Comparison

In [92], the NN output positive value indicates a good move, the larger the value, the better the move. In [36], one NN previously trained from a database of Go expert players' matches, tries to imitate local Go shapes; another NN estimates safely stones and a third NN tries to find potential unoccupied territory. In [109], an NN integrates local information across the board in all directions and produces outputs that represent ownership probabilities for identifying local territory to be occupied. In [21], an NN is used as the evaluation function of the leaf nodes in a game tree, with zero expertise involved. In contrast to our approach, the use of an NN is for pattern recognition of Go tactics. For this, an NN is trained with a set of *Don't care* patterns. Once the NN is trained, the board game is segmented into *windows view* size of 3×3 and 5×5 . Each *windows view* is encoded into a vector that serves as input to the NN, and the NN tries to find similarities with Go tactics, such as eyes, ladders and nets. Based on this recognition, a set of Go gaming actions is proposed. Table 4.5 summaries some aspects of the approaches based on NNs.

Table 4.5 Comparison of NN usage for playing Go.

| Approaches | Purpose | How is trained | Accuracy |
|-------------------|----------------|-----------------------|-----------------|
| N. Richard et al. | To define the | - | - |

| | | | |
|--------------------|---|--|------------|
| 1998 | next move | | |
| F. Dahl 2001 | Detect local shape, territorial and safe group | 400 games records | - |
| L. Wu et al. 2008 | Identify local territory ownership probabilities | From a dataset played by human | - |
| X. Cai et al. 2010 | Use to approximate the board evaluation function | Records of professional players | - |
| Own approach | Tactics patterns recognition to use on offensive / defensive. | Using a set of don't care patterns of eyes, ladders and nets forms | Above 70 % |

A comparative review of the NN-based approaches for automated Go players is shown in Table 4.6. The use of NNs for pattern recognition is essential in our proposal.

Table 4.6 Analytical comparison among different NN approaches.

| Go game proposal / Features | Our approach | N. Richard et al. 1998 | F. Dahl 2001 | L. Wu et al. 2008 | X. Cai et al. 2010 |
|--|---------------------|------------------------|--------------|-------------------|--------------------|
| Plays in small board size | ✓ | ✓ | ✓ | ✓ | ✓ |
| Plays in medium board size | ✓ | x | ✓ | ✓ | ✓ |
| Plays in large board size | ✓ | x | ✓ | ✓ | - |
| Board Segmentation | ✓ 3 x 3; 5 x 5 Size | x | x | ✓ 3 x 3 Size | ✓ |
| Multi-Players including | ✓ 4 | ✓ 2 | ✓ | - | - |
| FSM and formal languages Modeling | ✓ | x | x | x | x |
| Diversity strategies and tactics including | ✓ | ✓ | ✓ | ✓ | ✓ |
| Visual tactics recog- | ✓ | x | ✓ Detects | ✓ | x |

| | | | | | |
|-------------------------------|---|---|--------------------|---|---|
| inition | | | good or bad shapes | | |
| Visual strategies recognition | ✓ | x | x | x | x |

Efficiency [75] is the correct way to use the available resources for doing a task, measured by runtime. In our case, this is the time taken for training the net and simulating an entire match game; a reduced runtime implies more efficiency.

Efficacy [75] is the ability to achieve the desired goals or the realization of the activities to reach the goals, in our case is measured by the number of frequent similar patterns obtained. See Table 4.7 on for a comparison of efficiency and efficacy of the various NN Go automation approaches.

Table 4.7 Efficacy and Efficiency reported on the NN approaches.

| Issue / Feature | | Own approach | Richard et al. 1998 | F. Dahl 2001 | L. Wu et al. 2008 | Cai et al. 2010 |
|-----------------|----|---|--|--------------|--|-----------------|
| Efficiency | NN | 19 x 19 board size, each net training spends \approx 10 seconds | 9 x 9 board size, training spends 5 days, and "the training times increase with BS quite rapidly" [92] | - | For training a 9 x 9 board size is $O(N^4)$ and for future 19 x 19 could take months | - |
| Efficacy | NN | Net accuracy is above 70%. | - | - | "NN can learn territory predictions fairly well" [109] | - |

SANE [92] plays Go on a small board, cannot perform pattern recognition and needs 100 to 1000 simulations to achieve a 75% win rate over an adversary. Other approaches that use NNs, but do not present statistical results of accuracy are given in [36] [109] [21].

The results show that the strategic reasoning based on pattern recognition during the early and middle stages is appropriate because it allows the deployment of the strategically suitable moves through the deployment of previously known effective Go tactics and strategies. However, as the performance of this method reduces, MCTS is able to replace it and select moves based on the remaining free positions on the board. In the latter stages of a Go match, the switch to MCTS becomes an efficient option, because the size of the search space has become small and automated pattern recognition is too difficult to perform with the complex patterns on the board and the few remaining free board positions. Actually, in the late Go match stage, the deployment of *a-priori*-known strategies is difficult because the free board positions are too restrictive; few board spaces make it difficult to deploy strategies. Under this circumstance, the gaming method is by performing an MCTS evaluation to play any of the remaining free board positions. Therefore, in the early and middle match stages when few board points are occupied the Go *a-priori-knowledge*-based strategies are easy to deploy. At the end of a Go match, the use of MCTS is better for determining a move based on the empty board positions.

4.8 Gaming Discussion

The relevant advances by Monte-Carlo Tree Search applied to overcome the Go gaming huge complexity should be complemented to achieve to beat Go top level master people. Methods focused on simulation-based search algorithm [20, 47, 48] behaves very random in the early Go match stage so produce high search complexity in order to choice the next Go moves, because the huge set of board free positions at this match step. In contrast, by using pattern recognition and *a priori* known movements, sets the bases for efficient Go strategies/tactics gaming. The Go game *is the long-term influence*

moves [47], and moves made in the beginning affect the outcome of the later moves, so the relevance on doing right decisions in the early Go match stage. MCTS is particularly free from expert knowledge and from tactical solving guidance [56], and memory of previous game moves is based on a huge amount of simulations [47, 48], that results strongly time expensive in the early stage match. Furthermore, in specific situations it prevents to identify any right move since the lack in the use of a tactical search; it needs too many simulations per move for achieving an apt gaming move [33, 56]. Hence, an *a priori* knowledge-based method to movement election is smart at this step.

The method proposed in [107] for adversarial planning technique has some similarities to our proposal since both focus on building a sequence of planned actions such as eye, nets, ladders, capture, invasion, among others.

But our proposal uses NNs for pattern recognition to guide the sequence of planning actions. In contrast to [107] that uses a Hierarchical Task Network to build a tree search, where each node represent a plan for achieving the abstract goals and also each node in the search tree has one branch for each way the system suggests to further refine the plan.

Analysis on Go gaming automation from the complex network approach, like the one of the World Wide Web, focuses on the non-trivial topology of the network that results from a Go match [49]; the construction of a directed network given a suitable definition of related tactical Go gaming moves. By mining database matches of master level games, this approach discovered the similar patterns arising during the early stages of a Go match. In [34], the proposal for two dynamic randomization techniques is given: one for the parameters and the other for a hierarchical move generator.

The similarity between fractal formation and Go gaming patterns, is the diversity of the complex forms involved. However, a fractal formation follows a predetermined and regular pattern, but no previous regular pattern is followed by playing Go. Actually, the eyes, ladder and nets patterns are all obtained by ongoing strategies pertinent to each Go match.

Complex pattern recognition is present in Bioinformatics, which is devoted to computer and information analysis and the management of data on biological processes [58, 93,

108], particularly in determining or classifying molecular or tissue patterns as equivalent or related to some extent. Pattern recognition for Go gaming and Bioinformatics processes may advance in parallel from now on.

In computer complexity theory [98], the problems pertain to specific complexity classes by regarding certain characteristics: one major is *time*, which refers to the number of execution steps that an algorithm used to solve a problem; the other main complexity character is *space*, which refers to the amount of memory used to deal with a problem. Some complexity classes are P, NP, PSPACE, and EXPTIME. As a result of some complexity analyses of Go gaming, *experts* of the area claim that Go gaming belongs to EXPTIME-complete game [54], because it is an *unbounded two-player game*. *Unbounded games* are those in which there is no restriction on the number of moves that can be made. However, Go seems to be a bounded game because in each move a stone is placed, but there exist *capturing moves* that reopen spaces on the board. Papadimitriou [86], in one of his analyses, concluded that Go is a PSPACE-complete game.

Actually, being aware of what the adversary is doing helps to formulate defensive actions that inhibit her offensive actions. The inspired thinking that humans are capable of, as a result of observing the decisions other people make, applies in Go gaming through performing pattern recognition to decide on the next offensive/defensive move. The proposed NNs recognize forms that are Go tactics patterns and therefore, give relevant information to strategic decision making during the early and middle stages of a Go match.

4.9 Conclusion

We proposed the use of NNs for pattern recognition during the early and middle steps of a Go match; the expert's *a-priori-knowledge* for pattern recognition of eyes, ladders and nets is efficiently translated by means of NN for Go gaming automation. Based on this pattern recognition, defensive/offensive movements, such as those involved in complex Go gaming moves, are available to build up and apply during the middle steps of the match. On the other hand, during the latter stages of the game, the use of MCTS is appropriate because of the difficulty of performing *a-priori-knowledge* strategic gaming. A

relevant discussion of Go gaming analysis in a perspective of complex networks and fractals was introduced, and a mathematical modeling of a Go game was presented.

5

General Conclusion

5.1 Conclusions

The multi-player game modeling is of high complexity, and the strategic analysis of these sports should include a huge amount of parameters for fairly automated decision-making support.

The study of strategies in multi-player games such as Baseball and American Football is highly relevant so that it deserves an analysis like the one presented in this research work, since the selection of the players' strategies is a complex task therefore it needs a complex analysis of the match situations, thus proposing a plan of actions in order to increase the team expectations of win.

One contribution of this research work is the use of NE and/or the PE for the selection of the strategies of the players and Hungarian algorithm for team formation. This proposed methodology was tested through computer simulations of Baseball and AF matches, using real data of teams from the MLB and NFL.

In the top strategic game of Baseball, team's cooperation is essential for success. The usage of combined NE-PE strategy profile leads a Baseball team to choice strategies for building the match victory in such a way that every player's decision is by including from the other's ones. The combined NE-PE, as the applied strategy from the team is losing, pushes to close or overcome the match score to the current advantaging team; this gaming style mixing Nash individual-self-centered strategy with a Pareto collective-

optimal cooperation strength the team gaming. The assignment problem is a complex task even in sport like Baseball. In this thesis, we used the HA for assigning a set of players to a set of Baseball positions in order to improve the team performance and for coordination NE and PE as strategic analysis.

AF is one of the top sport games, the analysis of strategies is essential for a team success. In this thesis was presented a formal modeling of American Football using a formal grammar and finite state machine. The perspective of cooperation/non-cooperation influences the team performance. The teams performance using different methods of selection strategies were analyzed concluded that, the use of the NE for selection of strategies showed better team performance than that based on PE.

Moreover, the results from computer simulations of Baseball matches show that, although the usual *non-cooperative* qualification to NE, it is a relative adjective, up to the real circumstance. In the context of a Baseball match, with several parameters out of the players' and manager's control, NE allows identify strategy profiles for effective cooperation in real circumstances of Baseball gaming. The use of NE prevents to try plays or strategies with low statistical occurrence, so to avoid the risk to lose score points. It means that NE strategy profiles frequently include plays and strategies with higher statistical occurrence, so they are more feasible in real circumstances of matches. Furthermore, NE, by avoiding the risk to lose score points induces an effective cooperation for a team. On the other hand, PE formal account, it induces to choose the theoretically optimum strategy profiles. We observe that the best plays and strategies have low statistical occurrence, so few time to be practiced in real Baseball gaming circumstances. The Pareto efficient strategy profiles are less likely to occur than the Nash ones. Strategies in Pareto efficient profiles may be the most profitable but their probability of occurrence is low, and it moderates the use of Pareto efficiency to identify circumstances of cooperation in real circumstances of Baseball gaming.

The Go game is a top complex board game and currently, the deployment of learning algorithms for Go gaming automation is a central challenge in computer and artificial intelligence sciences. We proposed the use of NNs for pattern recognition during the early and middle steps of a Go match; the expert's *a-priori-knowledge* for pattern recognition

of eyes, ladders and nets is efficiently translated by means of NN for Go gaming automation. Based on this pattern recognition, defensive/offensive movements, such as those involved in complex Go gaming moves, are available to build up and apply during the middle steps of the match. On the other hand, during the latter stages of the game, the use of MCTS is appropriate because of the difficulty of performing *a-priori-knowledge* strategic gaming.

5.2 Summary of Contributions

In this section, we describe the major contributions of this doctoral work.

5.2.1 Methods

- I. Finite state machines and formal languages for Baseball.
- II. Finite state machines and formal languages for American Football.
- III. Finite state machines and formal languages for the game of Go.
- IV. Selection of strategies by using Nash equilibrium.
- V. Selection of strategies by using Pareto efficiency.
- VI. Selection of strategies by using mix of both methods for Baseball and American Football gaming.
- VII. Selection of strategies based on tactics pattern recognition using neural networks and based on Monte Carlo tree search.

5.2.2 Products

Software

- I. Baseball simulator with a module for the selection of strategies (using Nash equilibrium and / or Pareto efficiency).
- II. American Football simulator with a module for the selection of strategies (using Nash equilibrium and / or Pareto efficiency).

- III. An automated Go player, a graphic interface.

Papers

III. Published papers:

Journal papers

3. Matías Alvarado and Arturo Yee, "Nash equilibrium for collective strategic reasoning", **Expert System with Application**, 2012. **39**(15). Indexing in SCISEARCH, Science Citation Index, Scopus.
4. Matías Alvarado, Arturo Yee and Germinal Cocho, "Simulation of baseball gaming by cooperation and non-cooperation strategies", **Computación y Sistemas**, 2014. **18**(4). Indexing in CONACYT Index of Excellence of Mexican Journals, Scopus, Redalyc, E-Journal, e-revist@s, Latindex, Biblat, Periodica, DBLP, and SciELO.

Congress papers:

7. Arturo Yee Rendón and Matías Alvarado, "Formal language and reasoning for playing Go", Proceedings of LANMR'11, published, 2011.
8. Arturo Yee and Matías Alvarado, "Pattern Recognition and Monte-Carlo Tree Search for Go Gaming Better Automation", Proceedings of IBERAMIA 2012, LNAI 7637, 2012.
9. Matías Alvarado, Arturo Yee and Jesús Fernández, "Simulation of American Football Gaming", Advances in Sport Science and Computer Science, (ISSN: 1743- 3517), 2014.
10. Arturo Yee, Reinaldo Rodriguez and Matías Alvarado, "Analysis of Strategies in American Football Using Nash Equilibrium", the proceedings of AIMS 2014, LNCS 8722, 2014.
11. Arturo Yee and Matías Alvarado, "Methodology for the Modeling of Multi-Player Games", Proceedings of the 18th International Conference on Computers (part of CSCC '14), 2014.

12. Arturo Yee and Matías Alvarado, "Well-Time pattern recognition in Go gaming automation", Mathematical Methods and Computational Techniques in Science and Engineering, 2014.

IV. Conferences participations:

1. "Formal language and reasoning for playing Go". Seventh Latin American Workshop on logic/languages, Algorithms and New Methods of Reasoning, 2011.
2. "Razonamiento estratégico mediante Equilibrio de Nash en el juego de Béisbol". National Day of logic, Benemérita Universidad Autónoma de Puebla, México, 2011.
3. "Pattern Recognition and Monte-Carlo Tree Search for Go Gaming Better Automation". The 13th edition of the Ibero-American Conference on Artificial Intelligence IBERAMIA, Cartagena de Indias, Colombia, 2012.
4. "Simulation of American Football Gaming". 2013 International Conference on Sport Science and Computer Science, Hong Kong, China, 2013.
5. "Methodology for the Modeling of Multi-Player Games", The 18th International Conference on Computers (part of CSCC '14), Santorini, Grecia, 2014.
6. "Well-Time pattern recognition in Go gaming automation", Mathematical Methods and Computational Techniques in Science and Engineering, Atenas, Grecia, 2014.

V. Submitted Papers

5. Arturo Yee, Matías Alvarado and Germinal Cocho, "Simulation on best team formation and selection of strategies for baseball gaming", 2015, (under-review).
6. Arturo Yee and Matías Alvarado. "Simulation of strategic choices in an American football game", 2015, (under-review).
7. Matías Alvarado, Carlos Villareal, Sergio Camposortega and Arturo Yee, "Ising model for computer Go", 2015, (under-review).
8. Arturo Yee and Matías Alvrado, "Mathematical modeling and analysis of learning techniques for the game of Go", 2015, (under-review).

Patents

1. Simulador y módulo para la selección de estrategias al jugar Beisbol (expediente **MX/a/2015/002133**).

5.2.3 Impact in Media

Table 5.1 shows the impact of the newsletter title: "*Desarrolla Cinvestav simulador para preparar estrategias en béisbol*", this information was provided by Luisa Miranda Barbotó department chair of *comunicación, atención a medios*, Cinvestav.

Table 5.1 Information related to the impacts of newsletter.

| Title | Media name | Web address |
|---|----------------------|---|
| Cinvestav desarrolla simulador para preparar estrategias de beisbol | Alianza Tex | http://www.alianzatex.com/nota.php?nota=N0027855 |
| Juegan matemáticos al béisbol | AM | http://www.am.com.mx/notareforma/13143 |
| Diseña Cinvestav simulador para preparar estrategias de beisbol | Avance y Perspectiva | http://avanceyperspectiva.cinvestav.mx/4102/disena-cinvestav-simulador-para-preparar-estrategias-en-beisbol |
| Desarrollan simulador para preparar estrategias en beisbol | El Imparcial | http://www.elimparcial.com/EdicionEnLinea/Notas/CienciayTecnologia/10022014/807021-Desarrollan-simulador-para-preparar-estrategias-en-beisbol.html |
| Juegan matemáticos al béisbol | El Norte | http://www.elnorte.com/ciencia/articulo/786/1571804/ |
| Desarrolla Cinvestav simulador para preparar estrategias en beisbol | En Directo | http://endirecto.mx/?p=141471 |
| Desarrollan simulador para preparar estrategias en beisbol | Frontera Ensenada | http://www.fronteraensenada.info/EdicionEnLinea/Notas/CienciayTecnologia/10022014/807021-Desarrollan-simulador-para-preparar- |

| | | |
|--|----------------------------|---|
| | | estrategias-en-beisbol.html |
| Desarrollan simulador para preparar estrategias en beisbol | Frontera.info | http://www.frontera.info/EdicionEnLinea/Notas/CienciayTecnologia/10022014/807021-Desarrollan-simulador-para-preparar-estrategias-en-beisbol.html |
| Diseña Cinvestav simulador para preparar estrategias de beisbol | Investigación y Desarrollo | http://www.invdes.com.mx/ciencia-mobil/4076-disena-cinvestav-simulador-para-preparar-estrategias-en-beisbol |
| Desarrollan simulador para preparar estrategias en beisbol | La Crónica.com | http://www.lacronica.com/EdicionEnLinea/Notas/CienciayTecnologia/10022014/807021-Desarrollan-simulador-para-preparar-estrategias-en-beisbol.html |
| Desarrollan simulador para preparar estrategias en beisbol | Pop Buzz | http://www.popbuzz.me/mx/p/3008606/ |
| Juegan matemáticos al béisbol | Reforma | http://www.reforma.com/ciencia/articulo/730/1459397/ |
| Científicos mexicanos obtienen algoritmo capaz de ser manager de beisbol | Zona Franca | http://zonafranca.mx/cientificos-mexicanos-obtienen-algoritmo-capaz-de-ser-manager-de-beisbol/ |
| Desarrollan simulador para preparar estrategias en beisbol | Beisbol Sinaloa.com | http://www.beisbolsinaloa.com/index.php?option=com_content&view=article&id=13850%3Adesarrollan-simulador-para-preparar-estrategias-en-beisbol&catid=103%3Aotras-noticias&Itemid=130 |
| Diseña Cinvestav simulador para preparar estrategias de beisbol | Caribe Noticias | http://caribenoticias.com/2014/02/11/disena-cinvestav-simulador-para-preparar-estrategias-en-beisbol/ |
| Desarrolla Cinvestav simulador para preparar estrategias en beisbol | Conversión 21 | http://www.conversion21.com/index.php/programas-mainmenu-30/6924-desarrolla-cinvestav-simulador-para-preparar-estrategias-en-beisbol |
| Desarrollan simulador para preparar estrategias en beisbol | Enteradísimo | http://www.enteradisimo.com/noticia/1405560/desarrollan-simulador-para-preparar-estrategias-en-beisbol |
| Desarrolla Cinvestav simulador para preparar estrategias en beisbol | Gaiabit | http://www.gaiabit.com/category/cultura-y-salud/ |

| | | |
|---|-------------------|---|
| Play ball | La Crónica de Hoy | |
| Desarrollan simulador para preparar estrategias en beisbol | La Jornada.net | http://www.lajornadanet.com/diario/archivo/2014/febrero/11/11.php |
| Diseña Cinvestav simulador para preparar estrategias de beisbol | Tiempo en línea | http://www.tiempoenlinea.com.mx/index.php/using-joomla/extensions/components/content-component/article-categories/83-demo/news/tech/1650-disena-cinvestav-simulador-para-preparar-estrategias-de-beisbol |
| IPN desarrolla simulador que reemplaza a manager de beisbol | El Financiero | http://www.elfinanciero.com.mx/after-office/ipn-desarrolla-simulador-que-reemplazaria-a-manager-de-beisbol.html |
| IPN desarrolla simulador que reemplaza a manager de beisbol | En Tiempo real | http://entemporealmx.com/?p=174279 |

6

Future work

6.1 Potential Analysis by Product of our Work

In a broader perspective, our methodology for modeling complex games could analyze other types of complex problems such as:

- Physics aspects.
- Social dilemmas.

Although this is an idea “*thinking out loud*”, it could be interesting analyze some aspects of these problems and observe how to formulate formal models based on our work. Of course, to take any of the mentioned projects we require collaboration with experts on these areas.

6.2 Ising Model for Computer Go

In this section, we study a computational algorithm based on an Ising model and common fate graphs to estimate the value of strategies construction and territory control in computer Go. According to the simple rules of Go game, each other adversary single black and white stones (atoms) joint on complex stones shapes, struggling for achieving

territory control. We define the energy of stones in any Go board configuration by means of the Ising model. The Ising model, originally formulated to describe thermodynamic properties of ferromagnetic systems, considers a set of dichotomy variables representing discrete states of magnetization of magnetic dipole moments in atoms (spins). This interactions and the action of external fields may induce phase transitions between emergent states characterized by diverse degrees of spin ordering. Our point is that the Ising model fine describes the relationships among Go allied stones or the fighting with adversaries, as a stochastic process among dichotomy variables. The relations between stones in the Go board are implemented by assigning spin variables representing stone colors. The Go phase transition is displayed when black (white) stones make board area dominance by a movement, overriding the white (black) stones after balance dominance on this board area. We define the energy of stones in a Go board configuration on the base of the strength of these stones arrangement, that is, on the strength of every stone and their relative position on board. At the match end the stones with the highest energy yields to the winner.

6.2.1 The Ising Model

Our key point is that the Ising model is useful to describe the tactics construction process, so the relationships among allied stones or the fighting with adversaries may be simulated by the dynamics of this model. In a Go match we consider that a phase transition is displayed when black (white) stones make board control overriding the white (black) ones, and a single color stones prevail over the board. This is observed in a Go match when, from an initial configuration where the board appearance is of equilibrium between players controlling partial territories, placement of a single stone leads to capture of adversarial stones and control of most of the formerly shared area. See Figure 6.1.

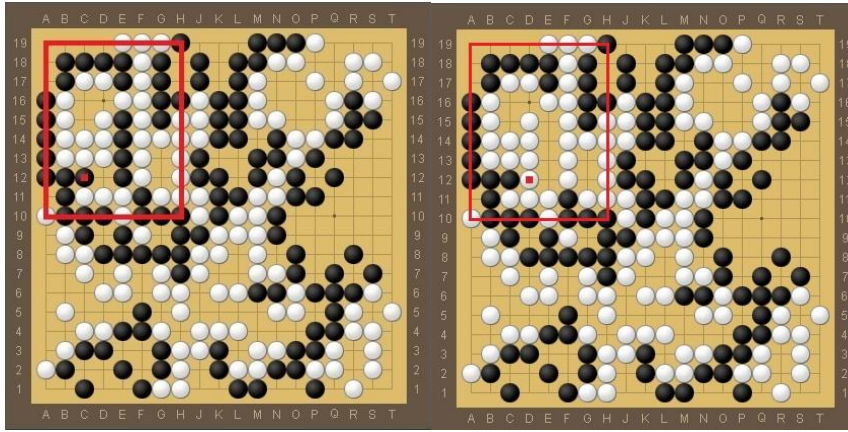


Figure 6.1 Phase transition by one stone placement on the Go board.

In the Ising model the atomic magnetic moments or spins are represented by dichotomic variables x_i representing the i th-spin state and may acquire any of two values, 1 or -1. The spins are arranged in an N -dimensional lattice; each spin interacts with neighboring spins or with external magnetic fields that tend to align them in the applied direction. The energy interaction is described by the Hamiltonian (6.1).

$$H = - \sum_{ij} w_{ij} x_i x_j - \mu \sum_i h_i x_i \quad (6.1)$$

Where w_{ij} represents the interaction strength between spin i and j , μ the magnitude of an external magnetic field, and h_i its relative contribution at site i . For a homogeneous external field $h_i = 1$. The Ising model has no phase transition in the 1D case, but displays phase transitions in 2D and higher dimensional cases. The alternative 2D phases of the Ising model described above are associated in our model to different stone configurations over the Go board.

The use of Ising model let us construct an algorithm for modeling tactics in computer Go gaming. In our model, the effective interactions between stones are represented by assigning a spin variable to each stone and its neighbors. The alternative spin states (1 or -1) are identified with the stone colors. The interactions can be graphically depicted in a lattice that allows the construction of a common fate graph [53], that yields a fair translation of the interaction dynamics in a Go match.


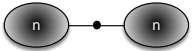
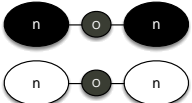
6.2.2 Go Tactics by Common Fate Graph

A common fate graph (CFG) is a form of representing any given state in a Go board. The CFG allows for grouping the stones in different shapes and relationships among them, by regarding the structure induced by the sequence of movements during Go gaming. CFGs have the following characteristics [53]:

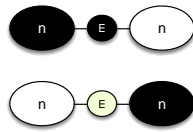
- Each stone on the board, single or compound, is a principal node in the CFG.
- Each principal node is labeled with the number of single stones (same color) that compose it.
- Each adjacent liberty to a stone on the board is considered a secondary node unlabeled and not colored.
- There are one or more edges between each principal node and one or more secondary nodes, depending of the liberties that have a compound stone.
- There is an edge whenever two different color compound stones are adjacent on the board

Definition of a CFG is extended. The representation of the stones and different patterns between them is shown in Table 6.1.

Table 6.1 Representation of the pattern of the stones for a CFG.

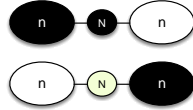
| Patron/move | Graphic Representation | Description |
|-----------------------|---|--|
| Compound stone |  | Compound stone of n single stones. |
| Single Liberty |  | Shared liberty between two compound stones, regardless of color. |
| Eyes |  | Liberty between same color stones. |

Ladder



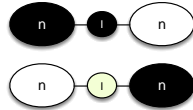
Liberty on a ladder between different colored stones. The color of liberty indicates the color of the stone that forms the ladder.

Net



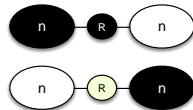
Liberty in a net between two stones of different colors. The color of liberty is the color of stone that form the net.

Invasion



Liberty to indicate that a stone makes invasion of territory control stone (s) opponents. The color of liberty is the ones of the stone making invasion.

Reduction



Liberty to indicate that a stone makes reduction of territory control stone (s) opponents. The color of liberty is the ones of the stone making reduction.

6.2.3 Example of CFG

We present an example for representing a given board configuration into a CFG. Figure 6.2 represents the stones configuration or state in a Go match played by **Murakawa** with black stones *versus* **Chao Chikun** with white stones in the 39th Japanese Kisei. We used the SGF of the match and built a CFG of the match. **Chao Chikun** who is one of the top players ranked in <http://www.go4go.net/>.

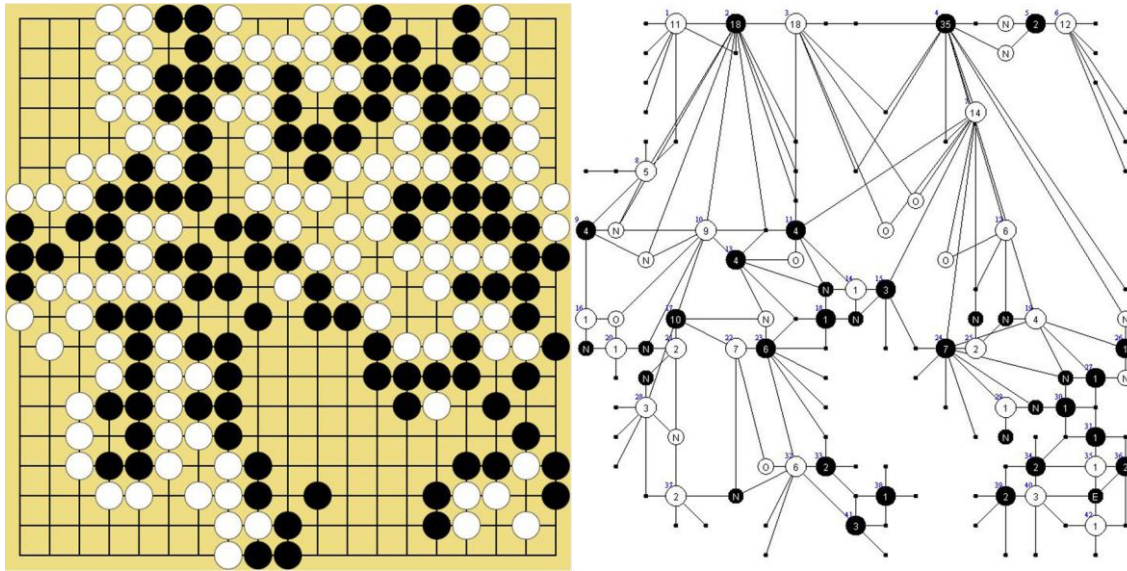


Figure 6.2 The representation of board configuration and CFG of **Murakawa** and **Chao Chikun**.

We remark the capacity to represent any Go board configuration by CFG representation, and this way the relationships among any Go configuration of stones that represent the sequences of tactics and strategies deployed during any Go match.

6.2.4 Go Modeling by Energy Function and CFG

In order to evaluate the energy associated to a given configuration we define an intrinsic strength function of a given stone to be introduced in the Ising Hamiltonian Eq. (6.1) defined above. We assume that the interaction strengths w_{ij} in the Ising model represent the ratio of union or repulsion between each pair of single or composite stones. The external field h_i represents the total number of liberties owned by stone i . The intrinsic strength of a single or compound stone is evaluated according to the following (6.2):

$$x_i = c_i(n_i + r_{eye}^{k_i}) \quad (6.2)$$

where n_i denotes the number of single stones that form a compound stone, r_{eyes} is a positive constant that represents the occurrence of an internal eye ($r_{eyes} = 0$ if there is no eye), k_i is the number of eyes inside a compound stone, and c_i is the color of the stone, 1 for white, and -1 for black. Observe that the term $r_0^{k_i}$ captures the importance of the number of eyes into one stone; in particular, $k_i = 2$ is for a non-capturable stone. On the other hand, the value of each w_{ij} depends on the liberties alongside possible paths joining stone i to j . They are affected by the presence and the intrinsic strength of adversary stones that may impede the $i - j$ connection. They are explicitly given by $w_{ij} = \sum_s r_t x_s^{(ij)}$, where $x_s^{(ij)}$ denotes the intrinsic strength of a stone s lying between i and j , and r_t represents the potency of a tactic pattern: eye (r_{eye}), net (r_{net}), ladder (r_{lad}), invasion (r_{inv}), reduction (r_{red}), and single liberty ($r_{sl} = 1$). These parameters form a total order $r_0 > r_{net} > r_{lad} > r_{inv} > r_{red} > r_{sl}$, determined by *a-priori* knowledge of Go tactics impact. A given tactic impact is estimated by an averaging procedure of observations of real matches between top-level players. A single eye is the top potency tactic for Go, followed by a net, a ladder, an invasion and a reduction.

The use of equations (6.1) and (6.2) permit to evaluate the energies of all the stones on the board for the sequence of states in a Go match, including the ultimate state that yields the final match. In summary, in order to calculate the energy of a given stone configuration we consider a complete combination of, the intrinsic stone strengths, the interaction strengths, the potency of tactics patterns, and the strategic movements.

The applying of Ising model to characterize a Go match let us identify phase transition-like phenomena occurring when a given color stone overrides the adversary one to achieve territory control in an isolated domain or the entire board. The evolution leading to territory control can be regarded as a phase transition that considers the following aspects: 1) The stronger a given color of compound stones the weaker the adversary ones; 2) in Ising model, the perturbation from an external field is considered. Likewise any stone in the board is affected from the global state in the board at any Go match step; 3) in Go game the combined power of allied stones it depends on the relative position and each other support arrangement over the board. Hence, Go gaming is fine de-

scribed by the Ising model. Moreover, the phenomenology of the game may be enlightened by the use of this classic mathematical model of complex interaction.

6.2.5 Experiments and Results

We present one set of experiments that use the present proposal to evaluate the energy of every configuration of stones in a Go board, each corresponding to each state of a Go match. Thus, the energy of the sequence of states, given by the sequence of made movements, is evaluated to decide what color stones has a better placement on the board in this state. The estimation from our proposal is compared against the human estimation. We use the information of Go matches reported at <http://www.go4go.net/>. Next figures represent the value of the Ising-model-based proposal on each move made by each player, blue line for blacks and red line for whites.

In Figure 6.3 we used the match played by **Murakawa** as black player versus **Chao Chikun** as white player in the 39th Japanese Kisei. During the first hundred of movements (states) an almost equal energy was present for black and white stones. During the movements 100 – 170 the energy values were separated and the separation becomes greater during the movements 171 – 230, but energy values become very close at the final movements. At the end, the total energy of black player is 667 and for the white player is 481. In this match, the result of the match reported in SGF is B+3.5.

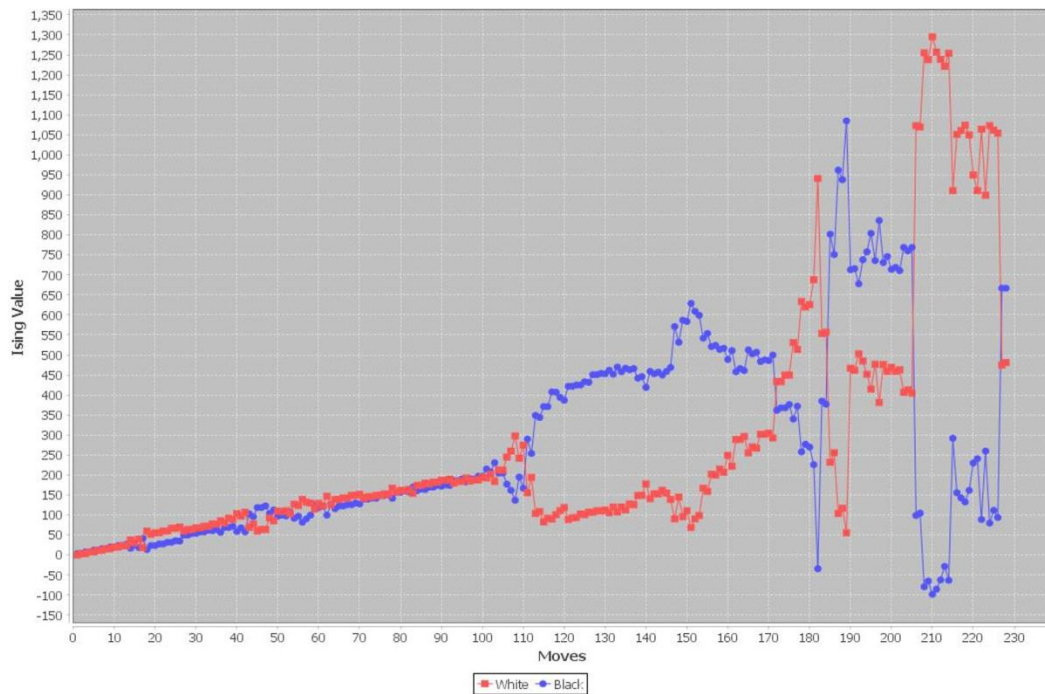


Figure 6.3 Stones energy from Ising-model-based proposal to the match of **Murakawa vs. Chao Chikun**

In Figure 6.4, we used a match played by **Lee Changho** as black player versus **Ryu Suhang 3p** as white player in Korean League 2014. They maintain a close fight during 70 off 100 movements, so their energies were almost the same in these movements. At move 73 a phase transition occurs and in this state and the next ones, the energy for blacks becomes each state greater than the energy for whites, until the end of the game. At the end, according to our proposal calculus the total energy of black stones is 424 and for the white player is -51. In this match, the result of the match reported in SGF is B+R.

In both of the described experiments the final result we got using our proposal is according to the one calculated by human Go players. We observe the closed fight during most of the movements in a match of top qualified Go players.

We made dozens of simulations with similar experiments to the above described. Next, we summarize the results on 30 ones of these matches, using the Ising-model-based proposal to evaluate the energy of stones configurations as well as the final result from

Go matches reported in <http://www.go4go.net/>. In 19 off 30 matches we got correct result, 8 off 30 we got close to correct result (minor error than 5%), and we got a bit non-correct result error, in a range of 6% - 10%, close to the correct match score. Please, see related material visiting http://delta.cs.cinvestav.mx/~matias/Teoria_Juegos/Go/EnergyFunction.

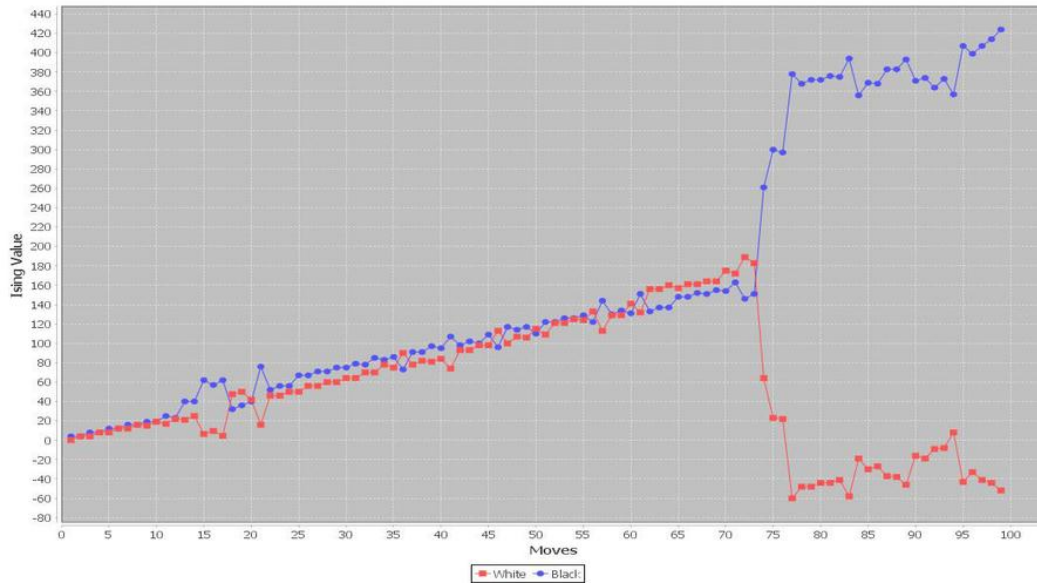


Figure 6.4 Stones energy from Ising-model-based proposal to the match of **Lee Changho** vs. **Ryu Suhang**.

Appendix 2.A

Here, we presented a deterministic FSM for Baseball gaming (see Figure 2.A.1). The FSM presented in (Figure 2.2) is non-deterministic, since in some states there are not transitions defined for every element of the alphabet; but for the smart modeling of Baseball gaming, this FSA works since it is able to recognize any string of the language generated by the Baseball formal grammar. The deterministic FSM that covers all the transitions given any state and any element from the alphabet follows (see Figure 2.A.1).

Let $(\Sigma, \hat{S}, s_0, \varphi, H)$ be a deterministic Baseball FSM such that:

- Σ is the alphabet, see Table 1.
- $\hat{S} = \{s, s_0, s_1, s_2, s_3, s_f\}$ is the set of states.
- $\varphi: \hat{S} \times \Sigma \rightarrow \hat{S}$ is the transitions function.
- $s_0 \in \hat{S}$ is the initial state.
- $H = \{s, s_0, s_f\} \subseteq \hat{S}$ is the set of halt states.

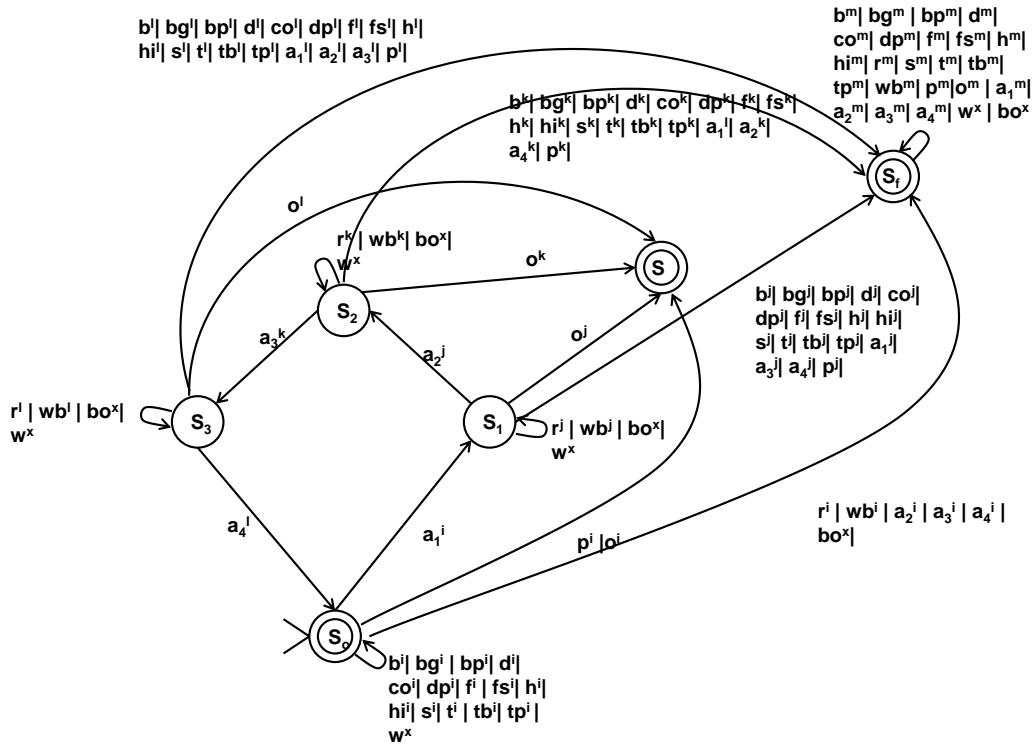


Figure 2.A.1. Deterministic baseball FSM

Bibliography

- [1] M.S. AL-Mutairi, Two-decision-maker cooperative games with fuzzy preferences, in: Proc. of the Industrial Engineering and Engineering Management (IEEM), 2010, pp. 6-12.
- [2] L.W. Alaways, M. Hubbard, Experimental determination of baseball spin and lift, J. Sport Sci., 19 (2001) 349-358.
- [3] M. Alvarado, A.Y. Rendón, Nash equilibrium for collective strategic reasoning, Expert Syst. Appl., 39 (2012) 12014-12025.
- [4] M. Alvarado, A. Yee, Baseball sacrifice play: towards the Nash Equilibrium based strategies, in: International Conference of 21st Game Theory Festival, Stony Brook University, 2010.
- [5] M. Alvarado, A. Yee, G. Cocho, Simulation of Baseball Gaming by Cooperation and Non-Cooperation Strategies, Computación y Sistemas, 18 (2014) 693 - 708.
- [6] L.V. Allis, Searching for Solutions in Games and Artificial Intelligence, in, University of Limburg, The Netherlands, 1994.
- [7] A.F.C. Association, Offensive Football Strategies, Human Kinetics, Champaign, IL, 2000.
- [8] R.D. Baker, I.G. McHale, Forecasting exact scores in National Football League games, Int. J. Forecasting, 29 (2013) 122-130.
- [9] S. Bandyopadhyay, P. Pathak, Knowledge sharing and cooperation in outsourcing projects - A game theoretic analysis, J. Decis. Support Syst., 43 (2007) 349-358.
- [10] S. Bandyopadhyay, J. Rees, J. Barron, Simulating sellers in online exchanges, J. Decis. Support Syst. , 41 (2006) 500-513.
- [11] D. Barker-Plummer, Turing Machines, The Stanford Encyclopedia of Philosophy, 2005.
- [12] D. Bell, H. Raiffa, A. Tversky, Decision Making, Cambridge University Press, 1999.
- [13] D.B. Benson, Life in the game of Go, Information Sciences, 10 (1976) 17-29.

- [14] V. Biló, A. Fanelli, M. Flammini, G. Melideo, L. Moscardelli, Designing Fast Converging Cost Sharing Methods for Multicast Transmissions, *J. Theor. Comput. Syst.* , 47 (2010) 507-530.
- [15] P.C. Bjarkman, *Diamonds Around the Globe: The Encyclopedia of International Baseball*, Greenwood Press, USA, 2004.
- [16] B. Bouzy, T. Cazenave, Computer Go: an AI Oriented Survey, *Artificial Intelligence*, 132 (2001) 39-103.
- [17] J.C. Bradbury, *The Baseball Economist: The Real Game Exposed*, The Penguin Group, New York, NY, 2008.
- [18] S.S. Britz, M.J.v. Maltitz, Application of the Hungarian Algorithm in Baseball Team Selection and Assignment, in: *Mathematical Statistics and Actuarial Science*, University of the Free State, 2011.
- [19] S.S. Britz, M.J.v. Maltitz, Application of the Hungarian Algorithm in Baseball Team Selection, in: *Department Mathematical Statistics and Actuarial Science*, University of the Free State, 2012.
- [20] C.B. Browne, E. Powley, D. Whitehouse, S.M. Lucas, P.I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, S. Colton, A Survey of Monte Carlo Tree Search Methods, *Computational Intelligence and AI in Games*, *IEEE Transactions on* 4(2012) 1-43.
- [21] X. Cai, G.K. Venayagamoorthy, D.C.W. II, Evolutionary swarm neural network game engine for Capture Go, *Neural Networks*, 23 (2010) 295-305.
- [22] W. Camp, C.S. Badgley, *American Football*, Createspace Independent Pub, 2009.
- [23] V. Capraro, M. Venanzi, M. Polukarov, N. Jennings, Cooperative Equilibria in Iterated Social Dilemmas, in: B. Vöcking (Ed.) *Algorithmic Game Theory*, Springer Berlin Heidelberg, 2013, pp. 146-158.
- [24] J. Clempner, A. Poznyak, Convergence method, properties and computational complexity for Lyapunov games, *Int. J. Appl. Math. Comput. Sci.*, 21 (2011) 349-361.
- [25] C.C. Coello, Gary B. Lamount, D.A. Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems*, Second ed., Springer, 2007.

- [26] C. Coen, Mixing rules: When to cooperate in a multiple-team competition, *Simul. Model. Pract. Th.*, 14 (2006) 423-437.
- [27] J.E. Cohen, Cooperation and self-interest: Pareto-inefficiency of Nash equilibria in finite random games, in: *Proc. of the Natl. Acad. Sci. USA*, USA, 1998, pp. 9724–9731.
- [28] A.M. Colman, *Game Theory and Experimental Games: The Study of Strategic Interaction*, Oxford, Pergamon Press, 1982.
- [29] H.W. Corley, P. Kwain, A Cooperative Dual to the Nash Equilibrium for Two-Person Prescriptive Games, *J. Appl. Math.*, 2014 (2014) 1-4.
- [30] R. Coulom, Computing Elo Ratings of Move Patterns in the Game of Go, *ICGA*, 30 (2007) 198-208.
- [31] G. Chalkiadakis, E. Elkind, M. Wooldridge, *Computational Aspects of Cooperative Game Theory*, Morgan and Claypool, 2011.
- [32] G. Charness, G.R. Fréchet, C.Z. Qin, Endogenous transfers in the Prisoner's Dilemma game: An experimental test of cooperation and coordination, *J. Game Econ. Behav.*, 60 (2007) 287-306.
- [33] G. Chaslot, S. Bakkes, I. Szita, P. Spronck, Monte-Carlo Tree Search: A New Framework for Game AI, in: *Proc. Artif. Intell. Interact. Digital Entert. Conf*, 2008, pp. 216-217.
- [34] K.-H. Chen, Dynamic randomization and domain knowledge in Monte-Carlo Tree Search for Go knowledge-based systems, *Knowledge-Based Systems*, 34 (2012) 21-25.
- [35] K. Chen, Z. Chen, Static analysis of life and death in the game of Go, *Inf. Sci. Inf. Comput. Sci.*, 121 (1999) 113-134.
- [36] F.A. Dahl, Honte, a go-playing program using neural nets, in: *Machines that learn to play games*, Nova Science Publishers, Inc., 2001, pp. 205-223.
- [37] S.J. Deutsch, P.M. Bradburn, A simulation model for American football plays, *Appl. Math. Model.*, 5 (1981) 13-23.
- [38] P. Dickson, *The New Dickson Baseball Dictionary*, 3rd ed., Harcourt Brace, USA, 1999.

- [39] A. Dornhaus, Finding optimal collective strategies using individual-based simulations: colony organization in social insects, *Math. Comp. Model. Dyn.*, 18 (2012) 25-37.
- [40] P. Drake, Y.-P. Chen, Coevolving partial strategies for the game of go, in: the International Conference on Genetic and Evolutionary Methods 2008, pp. 312-318.
- [41] P.K. Dutta, *Strategies and games: theory and practice*, Massachusetts Institute of Technology, USA, 1999.
- [42] P. Dütting, M. Henzinger, I. Weber, Sponsored search, market equilibria, and the Hungarian Method, *Inf. Process. Lett.*, 113 (2013) 67-73.
- [43] Elwyn R. Berlekamp, D. Wolfe, *Mathematical Go: Chilling Gets the Last Point*, A K Peters Ltd Massachusetts, 1997.
- [44] C. Fellows, Y. Malitsky, G. Wojtaszczyk, Exploring GnuGo's evaluation function with a SVM, in: proceedings of the 21st national conference on Artificial intelligence - Volume 2, AAAI Press, Boston, Massachusetts, 2006, pp. 1867-1868.
- [45] M. Gairing, T. Lucking, M. Mavronicolas, B. Monien, M. Rode, Nash equilibria in discrete routing games with convex latency functions, *J. Comput Syst. Sci.*, 74 (2008) 1199-1225.
- [46] D. Gale, L.S. Shapley, College Admissions and the Stability of Marriage, *The American Mathematical Monthly*, 69 (1962) 9-15.
- [47] S. Gelly, L. Kocsis, M. Schoenauer, M. Sebag, D. Silver, C. Szepesvari, O. Teytaud, The Grand Challenge of Computer Go: Monte Carlo Tree Search and Extensions, *Communication of the ACM* 55, 3 (2012) 106-113.
- [48] S. Gelly, D. Silver, Monte-Carlo tree search and rapid action value estimation in computer Go, *Artificial Intelligence*, 175 (2011) 1856-1875.
- [49] B. Georgeot, O. Giraud, The game of go as a complex network, *Europhysics Letters*, 97 (2012).
- [50] C. Gifford, *American Football Tell me about Sport*, Evans Publishing, London, U.K., 2009.
- [51] J. Goldberger, T. Tassa, A hierarchical clustering algorithm based on the Hungarian method, *Pattern Recogn. Lett.*, 29 (2008) 1632-1638.

- [52] A.J. Gonzalez, D.L. Gross, Learning tactics from a sports game-based simulation, *Int. J. Comput. Simul.*, 5 (1995) 127-148.
- [53] T. Graepel, M. Goutrié, M. Krüger, R. Herbrich, Learning on Graphs in the Game of Go, in: G. Dorffner, H. Bischof, K. Hornik (Eds.) *Artificial Neural Networks — ICANN 2001*, Springer Berlin Heidelberg, 2001, pp. 347-352.
- [54] R.A. Hearn, *Games, Puzzles, and Computation*, in: Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, 2006.
- [55] S. Hémon, M. Rougemont, M. Santha, Approximate Nash Equilibria for Multi-player Games, in: *Proc. of the 1st International Symposium on Algorithmic Game Theory*, 2008, pp. 267-278.
- [56] J.B. Hoock, L. Chang-Shing, A. Rimmel, F. Teytaud, W. Mei-Hui, O. Teytaud, Intelligent Agents for the Game of Go, *Computational Intelligence Magazine, IEEE*, 5 (2010) 28-42.
- [57] J.E. Hopcroft, J.D. Ullman, *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley, Cambridge, 1979.
- [58] M. Hue, M. Riffle, J.-P. Vert, W. Noble, Large-scale prediction of protein-protein interactions from structures, *BMC Bioinformatics*, 11 (2010) 1-9.
- [59] C.T.L. Janssen, T.E. Daniel, A Decision Theory Example in Football, *Decision Sci*, 15 (1984) 253-259.
- [60] L. Jianming, Z. Difei, L. Rui, Improve Go AI based on BP-Neural Network, in: *Cybernetics and Intelligent Systems, 2008 IEEE Conference on*, 2008, pp. 487-490.
- [61] T. Jinji, S. Sakurai, Y. Hirano, Factors determining the spin axis of a pitched fastball in baseball, *J. Sport Sci.*, 29 (2011) 761-767.
- [62] T. Kar Bin, J. Teo, P. Anthony, Multi-objective Evolution of Neural Go Players, in: *Digital Game and Intelligent Toy Enhanced Learning (DIGITEL), 2010 Third IEEE International Conference on*, 2010, pp. 46-53.
- [63] A. Kelly, *Decision Making using Game Theory An introduction for managers*, Cambridge University Press, 2003.
- [64] J. Kim, *Learn to Play Go*, Good Move Press, New York, 1994.

- [65] K.A. Konrad, Altruism and envy in contests: An evolutionarily stable symbiosis, *Soc. Choice Welfare*, 22 (2004) 479-490.
- [66] H.W. Kuhn, The Hungarian method for the assignment problem, *Naval Res. Logist. Quart*, 2 (1955) 83-97.
- [67] Y. Lei, Z. Xiaojin, D. Chunni, L. Jingao, Solving winning probability problems in Go, in: *Advanced Computer Control (ICACC), 2010 2nd International Conference on*, 2010, pp. 477-480.
- [68] G.R. Lindsey, An Investigation of Strategies in Baseball, *J. Oper. Res.*, 11 (1963) 477-501.
- [69] R.J. Lipton, E. Markakis, A. Mehta, Playing large games using simple strategies, in: *Proc. of the 4th ACM Conference on Electronic commerce*, 2003, pp. 36-41.
- [70] C. MacMahon, J.L. Starkes, Contextual influences on baseball ball-strike decisions in umpires, players, and controls, *J. Sport Sci.*, 26 (2008) 751-760.
- [71] J. McCarthy, AI as sport, *Science*, 276 (1997) 1518-1519.
- [72] A.G. McGrew, M.J. Wilson, *Decision making: approaches and analysis*, Manchester University Press, 1982.
- [73] J. McMillan, *Games strategies and managers*, Oxford University Press, 1996.
- [74] K. Miettinen, *Nonlinear Multiobjective Optimization*, Springer US, 1998.
- [75] F.P. Miller, A.F. Vandome, J. McBrewster, *Computational Complexity Theory*, VDM Publishing House Ltd., Saarbrücken, Germany, 2009.
- [76] H. Mintzberg, J.B. Quinn, *The Strategy Process: Concepts, Context, Cases*, 2nd ed., Prentice Hall, Englewood Cliffs, N. J., 1991.
- [77] MLB, in, 2012.
- [78] M. Müller, Computer Go, *Artificial Intelligence*, 134 (2002) 145-179.
- [79] J. Munkres, Algorithms for the Assignment and Transportation Problems, *J. Soc. Ind. Appl. Math.*, 5 (1957).
- [80] V. Nanduri, T. Das, A Reinforcement Learning Algorithm for obtaining Nash Equilibrium of Multi-player Matrix Games, *J. IIE Trans.*, 41 (2009) 158-167.
- [81] J. Nash, Non-Cooperative Games, *The annals of Mathematics*, 54 (1951) 286-295.

- [82] J.v. Neumann, O. Morgenstern, *Theory of Game and Economic Behavior*, 2nd ed., Nueva Jersey: Princenton University Press, 1944.
- [83] NFL, in.
- [84] E.H.J. Nijhuis, *Learning Patterns in the Game of Go*, in: *Artificial Intelligence*, Universiteit van Amsterdam, Amsterdam, 2006.
- [85] N. Nisan, T. Roughgarden, E. Tardos, V.V. Vazirani, *Algorithmic Game Theory*, Cambridge University Press, 2007.
- [86] C.H. Papadimitriou, *Computational Complexity*, Addison Wesley, Reading, Massachusetts 1993.
- [87] C.H. Papadimitriou, T. Roughgarden, *Computing equilibria in multi-player games*, in: *Proc. of the 16th SODA*, 2005, pp. 82-91.
- [88] V. Pareto, *Cours d'économie politique*, F. Rouge Lausanne, 1-2 (1896).
- [89] N.G. Pavlidis, K.E. Parsopoulos, M.N. Vrahatis, *Computing Nash equilibria through computational intelligence methods*, *J. Comput. Appl. Math.*, 175 (2005) 113 -136.
- [90] M. Pratola, T. Wolf, *Optimizing GoTools' Search Heuristics using Genetic Algorithms*, *International Computer Games Association*, 26 (2003) 28-49.
- [91] F.V. Redondo, *Economy and Games*, Antoni Bosh, Spain, 2000.
- [92] N. Richards, D.E. Moriarty, R. Miikkulainen, *Evolving Neural Networks to Play Go*, *Applied Intelligence*, 8 (1998) 85-96.
- [93] M.D. Ritchie, A.A. Motsinger, W.S. Bush, C.S. Coffey, J.H. Moore, *Genetic programming neural networks: A powerful bioinformatics tool for human genetics*, *Appl. Soft Comput.*, 7 (2007) 471-479.
- [94] J.E. Roemer, *Kantian Equilibrium*, *Scand. J. of Econ.*, 112 (2010) 24.
- [95] J.E. Roemer, *Kantian Optimization, Social Ethos, and Pareto Efficiency*, *Cowles Foundation Discussion Paper No. 1854* (2012).
- [96] D. Ross, *Game Theory*, The Stanford University, 2008.
- [97] F.I. Roy, *The rise and fall of collective strategies: a case study*, *Int. J. Entrepreneurship and Small Business*, 5 (2008) 124-142.
- [98] Sanjeev Arora, B. Barak, *Computational Complexity: A Modern Approach*, Cambridge University Press, Cambridge, 2009.

- [99] Y. Shi-Jim, Y. Tai-Ning, C. Chang, H. Shun-Chin, Pattern Matching in Go Game Records, in: Innovative Computing, Information and Control, 2007. ICICIC '07. Second International Conference on, 2007, pp. 297-297.
- [100] K. Shiba, K. Mori, Detection of Go-board contour in real image using genetic algorithm, in: SICE 2004 Annual Conference, 2004, pp. 2754-2759 vol. 2753.
- [101] D. Silver, R.S. Sutton, M. M, #252, Iler, Temporal-difference search in computer Go, Mach. Learn., 87 (2012) 183-219.
- [102] D. Silver, R.S. Sutton, M. Müller, Temporal-difference search in computer Go, Mach. Learn., 87 (2012) 183-219.
- [103] M. Sipser, Introduction to Theory of Computation, 2nd ed., Thomson Course Technology, 2006.
- [104] C. Song, B.L. Boulier, H.O. Stekler, The comparative accuracy of judgmental and model forecasts of American football games, Int. J. Forecasting, 23 (2007) 405-413.
- [105] E.C.D. van der Werf, H.J. van den Herik, J.W.H.M. Uiterwijk, Learning to Estimate Potential Territory in the Game of Go, in: H.J. van den Herik, Y. Björnsson, N.S. Netanyahu (Eds.) Computers and Games, Springer, 2004, pp. 81-96.
- [106] T. Williams, Winning Strategies for Offense and Defense, Baseball's Best, 2005.
- [107] S. Willmott, J. Richardson, A. Bundy, J. Levine, Applying adversarial planning techniques to Go, Theor. Comput. Sci., 252 (2001) 45-82.
- [108] O. Wolkenhauer, D. Fell, P. De Meyts, N. Bluthgen, H. Herzel, N. Le Novere, T. Hofer, K. Schurrle, I. van Leeuwen, SysBioMed report: Advancing systems biology for medical applications, Systems Biology, IET, 3 (2009) 131-136.
- [109] L. Wu, P. Baldi, Learning to play Go using recursive neural networks, Neural Networks, 21 (2008) 1392-1400.
- [110] N. Yoshiaki, Strategic Concepts of Go, Ishi Press, Japan, 1972.
- [111] L. Zhiqing, L. Wenfeng, Contextual patterns and pattern collocations in the game of GO, in: Control and Decision Conference (CCDC), 2011 Chinese, 2011, pp. 2527-2532.