ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Διπλωματική Εργασία

Σχεδίαση και Ανάπτυξη Αυτόνομου Πράκτορα για το Επιτραπέζιο Παιχνίδι Mastermind

Αθανάσιος Μανεσιώτης

Εξεταστική Επιτροπή
Καθηγητής Μιχαήλ Γ. Λαγουδάκης (Επιβλέπων)
Αναπληρωτής Καθηγητής Γεώργιος Χαλκιαδάκης
Καθηγητής Μιχάλης Ζερβάκης

Χανιά, Μάρτιος 2023

TECHNICAL UNIVERSITY OF CRETE

SCHOOL OF ELECTRICAL AND

COMPUTER ENGINEERING

Diploma Thesis

Design and Development of Autonomous Agent for the Mastermind Board Game

Athanasios Manesiotis

Thesis Committee

Professor Michail G. Lagoudakis (Supervisor)
Associate Professor Georgios Chalkiadakis
Professor Michalis Zervakis

Chania, March 2023

ΠΕΡΙΛΗΨΗ

Η Τεχνητή Νοημοσύνη ασχολείται με τη σχεδίαση και την κατασκευή ευφυών οντοτήτων/μηχανών, οι οποίες είναι ικανές να εκτελέσουν λειτουργίες που αποδίδονται σε ανθρώπινη νοημοσύνη. Οι οντότητες/μηχανές που μπορούν να ενεργούν αποτελεσματικά και με ασφάλεια σε ένα ευρύ φάσμα νέων καταστάσεων καλούνται ευφυείς πράκτορες. Ένας ορθολογικός πράκτορας είναι ένας ευφυής πράκτορας, ο οποίος ενεργεί έτσι ώστε να μεγιστοποιεί το αναμενόμενο όφελος σε συνθήκες αβεβαιότητας, να επιλέγει δηλαδή ενέργειες που αναμένεται να μεγιστοποιήσουν το μέτρο της απόδοσής του. Τα επιτραπέζια παιχνίδια αποτέλεσαν και αποτελούν μια σημαντική περιοχή έρευνας για τη ανάπτυξη και δοκιμή εφαρμογών Τεχνητής Νοημοσύνης, με σημαντικές επιτυχίες και νίκες απέναντι στους ανθρώπους σε ανταγωνιστικά παίγνια δύο παικτών. Η παρούσα διπλωματική εργασία εστιάζει στην σχεδίαση και ανάπτυξη αυτόνομων πρακτόρων για το επιτραπέζιο παιχνίδι Mastermind. Οι πράκτορες μπορούν να ανταπεξέλθουν, τόσο στην κλασσική έκδοση του παιχνιδιού, όσο σε πιο σύνθετα επίπεδα δυσκολίας που αναπτύχθηκαν στο πλαίσιο της εργασίας. Η εργασία, πέρα από την ανάλυση και σύγκριση των υλοποιημένων πρακτόρων, παρέχει επίσης ένα φιλικό γραφικό περιβάλλον για τον χρήστη, προκειμένου να οπτικοποιούνται και να γίνονται άμεσα αντιληπτά τα αποτελέσματα των στρατηγικών των πρακτόρων. Επιπρόσθετα, δίνεται η δυνατότητα στο χρήση να παίξει μόνος του το παιχνίδι. Τόσο ο κώδικας για το γραφικό περιβάλλον, όσο και οι στρατηγικές που ακολουθούν οι πράκτορες, υλοποιήθηκαν σε γλώσσα Java, εκμεταλλευόμενοι τις δυνατότητες που παρέχει η συγκεκριμένη γλώσσα προγραμματισμού. Τα αποτελέσματα που εξήχθησαν είναι αρκετά ικανοποιητικά, καθώς στην κλασσική έκδοση του παιχνιδιού, ο προτεινόμενος πράκτορας με κάποιες από τις υλοποιημένες στρατηγικές εντοπίζει τον κρυφό χρωματικό κωδικό το πολύ σε πέντε (5) προσπάθειες.

ABSTRACT

Artificial Intelligence (AI) deals with the design and construction of intelligent entities/machines, capable of executing functions attributable to human intelligence. These entities/machines that can act efficiently and safely in a wide range of new situations are called intelligent agents. A rational agent is an intelligent agent, that can act in such way as to maximize the expected utility under conditions of uncertainly, i.e. to choose actions that are expected to maximize its performance metric. Board games have been an important area of research for the development and testing of AI applications, with significant successes and victories over humans in competitive two-player games. This diploma thesis focuses on the design and development of autonomous agents for the board game Mastermind. The agents can cope with both the classic version of the game and the more complex levels of difficulty developed within this thesis. In addition to the analysis and the comparison of the implemented agents, the thesis provides a user-friendly graphical interface for the user to visualize and immediately understand the results of the strategies of the agents. In addition, the user is given the opportunity to play the game on his own. Both the code for the graphical interface and the strategies followed by the agents were implemented in Java, taking advantage of the possibilities provided by this programming language. The extracted results are quite satisfactory, since in the classic version of the game, the proposed agent with some of the implemented strategies detects the hidden color code in a maximum of five (5) attempts.

ΕΥΧΑΡΙΣΤΙΕΣ

Πρωτίστως, θα ήθελα να ευχαριστώ θερμά την οικογένεια μου, για την στήριξη που μου παρείχε όλα αυτά τα χρόνια και για την δυνατότητα που μου προσέφερε να σπουδάσω στη σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Πολυτεχνείου Κρήτης εκπληρώνοντας ένα από τα όνειρά μου. Έπειτα, θα ήθελα να ευχαριστήσω τους φίλους και συμφοιτητές μου για την συμπαράσταση που μου παρείχαν, αλλά και για το όμορφο αυτό ταξίδι που πραγματοποιήσαμε παρέα. Τέλος, θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή μου κ. Μιχαήλ Λαγουδάκη για την αποδοχή και την εμπιστοσύνη που μου έδειξε, καθώς και για την καθοδήγησή του, χάρη στην οποία ολοκληρώθηκε επιτυχώς η παρούσα διπλωματική εργασία.

ΠΕΡΙΕΧΟΜΕΝΑ

1 ΕΙΣΑΓΩΓΗ

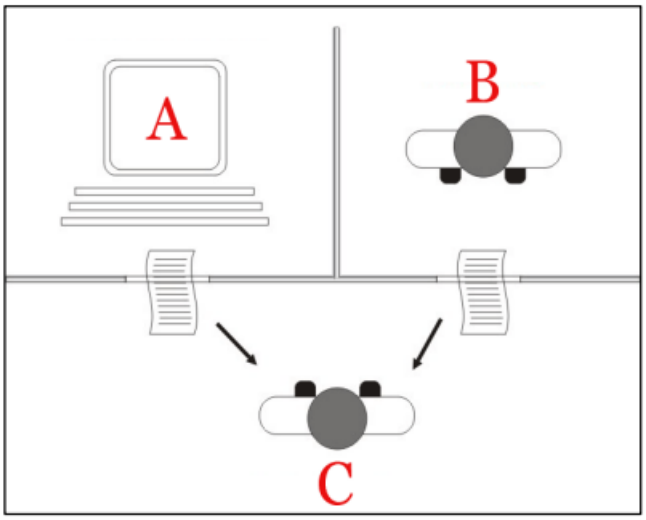
1.1 Τεχνητή Νοημοσύνη: Εισαγωγικές έννοιες

1.1.1 Τι είναι Τεχνητή Νοημοσύνη

Καλούμε το ανθρώπινο είδος homo sapiens (άνθρωπος ο σοφός), επειδή η νοημοσύνη είναι συνυφασμένη με τη ανθρώπινη υπόσταση. Η νοημοσύνη συνίσταται στην ικανότητα του ατόμου να μαθαίνει από την εμπειρία του, να προσαρμόζεται σε νέες καταστάσεις, να κατανοεί και να χειρίζεται αφηρημένες έννοιες και να χρησιμοποιεί τη γνώση για να ελέγξει το περιβάλλον του. Ουσιαστικά, αφορά την ικανότητα του ατόμου να αποκτά γνώσεις, δεξιότητες και τεχνικές επεξεργασίας πληροφοριών [1], [2].

Η Τεχνητή Νοημοσύνη (Artificial Intelligence) αποτελεί ένα από τους πιο σημαντικούς τομείς της τεχνολογίας, που ασχολείται με την κατανόηση και κατασκευή ευφυών οντοτήτων-μηχανών που μπορούν να υπολογίζουν πώς να ενεργούν αποτελεσματικά και με ασφάλεια σε ένα ευρύ φάσμα νέων καταστάσεων. Η Τεχνητή Νοημοσύνη (ΤΝ) περιλαμβάνει μεγάλο πλήθος ερευνητικών αντικειμένων όπως η αντίληψη και η συλλογιστική, απόδειξη θεωρημάτων, κατανόηση φυσικής γλώσσας, παίγνια στρατηγικής, μηχανική μάθηση (Machine Learning-ML), βαθιά μάθηση (Deep Learning-DL) κλπ. [3], [4].

Η αρχή της σύγχρονης έρευνας για την τεχνητή νοημοσύνη μπορεί να αναχθεί στον John McCarthy του MIT (Ινστιτούτο Τεχνολογίας της Μασαχουσέτης), ο οποίος εισήγαγε τον όρο «Τεχνητή Νοημοσύνη-ΤΝ» (Artificial Intelligence-AI) σε ένα συνέδριο στο Dartmouth College το 1956 [5]. Ωστόσο, αναφορά για την ΤΝ, γίνεται αρκετά χρόνια πριν στην εργασία του Alan Turing, "Computing Machinery and Intelligence", η οποία δημοσιεύθηκε το 1950, στην οποία ουσιαστικά τίθεται το ερώτημα «Μπορούν οι μηχανές να σκέφτονται;» και «Πότε θεωρείται μια μηχανή ευφυής;» Ο Turing παρουσιάζει μια δοκιμασία, γνωστή ως «Δοκιμασία Turing-Turing test», όπου ένας άνθρωπος-εξεταστής προσπαθεί να διακρίνει εάν μια απάντηση κειμένου προέρχεται από υπολογιστή (μηχανή) ή από άνθρωπο (Σχήμα 1.1). Εάν ο εξεταστής δεν καταφέρει να διακρίνει από που προέρχεται η απάντηση, τότε η μηχανή είναι ευφυής. Αν και το test έχει δεχθεί σημαντικές κριτικές από τη δημοσίευσή του, παραμένει ένα σημαντικό μέρος της ιστορίας της ΤΝ [6].



Σχήμα 1.1: Test Turing [7]

(Το Α αντιστοιχεί στον υπολογιστή, το Β στον άνθρωπο και το C στον εξεταστή)

Τα επόμενα χρόνια συντελέστηκε σημαντική πρόοδος όσον αφορά το πεδίο της ΤΝ. Πολλοί επιστήμονες και ερευνητές επικεντρώθηκαν στην αυτοματοποιημένη συλλογιστική και αξιοποίησαν την τεχνητή νοημοσύνη για την επίλυση αλγεβρικών προβλημάτων και την απόδειξη μαθηματικών θεωρημάτων. Ο Marvin Minsky (Πανεπιστήμιο Carnegie-Mellon) ορίζει την ΤΝ ως την κατασκευή μηχανών που εμπλέκονται σε εργασίες που απαιτούν υψηλού επιπέδου νοητικές διεργασίες, εάν εκτελούνταν από ανθρώπους όπως: αντιληπτική μάθηση, οργάνωση μνήμης και κριτική λογική [8], [9].

Πολλά χρόνια αργότερα, οι Stuart Russell και Peter Norvig στο βιβλίο τους “Artificial Intelligence: A Modern Approach”, ταξινομούν την ΤΝ σε τέσσερις πιθανές κατηγορίες με βάση τον ορθολογισμό και τη σκέψη [9]:

<u>Ανθρώπινη προσέγγιση:</u>

- Συστήματα που σκέφτονται όπως οι άνθρωποι
- Συστήματα που ενεργούν όπως οι άνθρωποι

<u>Ιδανική προσέγγιση:</u>

- Συστήματα που σκέφτονται ορθολογικά
- Συστήματα που ενεργούν ορθολογικά

Στους στόχους της ΤΝ είναι η ανάπτυξη μηχανών που θα μπορούν να σκέφτονται όπως οι άνθρωποι και να μιμούνται ανθρώπινες συμπεριφορές, συμπεριλαμβανομένης της αντίληψης, του συλλογισμού, της μάθησης, του προγραμματισμού, της πρόβλεψης, της λήψης αποφάσεων κ.λπ. [10]. Η ΤΝ θεωρείται σημαντική γιατί [11]:

- Αυτοματοποιεί την επαναλαμβανόμενη εκμάθηση και ανακάλυψη μέσω δεδομένων, εκτελώντας συχνές, μεγάλου όγκου, τυποποιημένες εργασίες. Το παραπάνω δεν αναιρεί το σημαντικό ρόλο των ανθρώπων, καθώς οι άνθρωποι ρυθμίζουν τα συστήματα και θέτουν τις σωστές ερωτήσεις.
- Προσθέτει νοημοσύνη σε υπάρχοντα προϊόντα/υπηρεσίες. Ενδεικτικά, αναφέρονται οι πλατφόρμες συνομιλίας, οι αυτοματισμοί, οι έξυπνες μηχανές, τεχνολογίες Industry 4.0 κλπ.
- Προσαρμόζεται μέσω αλγορίθμων προοδευτικής μάθησης, καθώς εντοπίζει δομή και κανονικότητες στα δεδομένα. Ένας ευφυής πράκτορας για παράδειγμα εκτός από το να μπορεί να μάθει μόνος του να παίζει σκάκι, μπορεί να μάθει να προτείνει πιθανούς παραλήπτες ηλεκτρονικού ταχυδρομείου.
- Αναλύει πολύ μεγάλο σύνολο δεδομένων (big data), με πολλά από αυτά να είναι αχαρακτήριστα (unlabeled).
- Επιτυγχάνει μεγάλη ακρίβεια χρησιμοποιώντας νευρωνικά δίκτυα πολλών επιπέδων π.χ. για αυτοκίνητα χωρίς οδηγό.

Παραδείγματα αξιοποίησης της ΤΝ σήμερα είναι: ρομποτικά οχήματα, κίνηση με ρομποτικά πόδια, αυτόνομος σχεδιασμός και χρονοπρογραμματισμός, μηχανική μετάφραση, αναγνώριση ομιλίας, συστάσεις, παιχνίδια, κατανόηση εικόνων, ιατρική, κλιματική επιστήμη κλπ. [3].

1.1.2 Τεχνητή Νοημοσύνη και Ορθολογική Σκέψη

Ακόμη και σήμερα οι επιστήμονες προσπαθούν να κατανοήσουν πώς σκέπτονται και ενεργούν τα άτομα, και ειδικότερα πώς ο ανθρώπινος εγκέφαλος αντιλαμβάνεται, κατανοεί, προβλέπει και χειρίζεται έναν κόσμο πολύ μεγαλύτερο και πιο πολύπλοκο από τον εαυτό του [3].

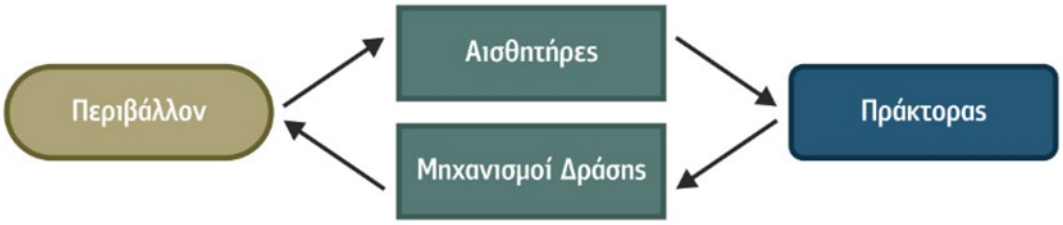
Ο Έλληνας φιλόσοφος Αριστοτέλης θεωρείται ο θεμελιωτής της Λογικής. Ήταν από τους πρώτους που επιχείρησαν να κωδικοποιήσουν τη «σωστή σκέψη», δηλαδή, αδιάψευστες διαδικασίες συλλογιστικής. Η εισαγωγή και η ανάλυση των συλλογισμών είναι

ίσως η μεγαλύτερη συμβολή του Αριστοτέλη στη φιλοσοφία. Η αριστοτελική λογική ξεκινά από την ανάλυση των απλών προτάσεων της γλώσσας. Μια απλή πρόταση της μορφής «ο Σωκράτης είναι φιλόσοφος», δηλαδή μια πρόταση που συνδέει ένα υποκείμενο με ένα κατηγορούμενο δίνοντάς μας μια πληροφορία, είναι το ελάχιστο στοιχείο της γλώσσας που παρουσιάζει λογικό και φιλοσοφικό ενδιαφέρον. Οι συλλογισμοί του παρείχαν πρότυπα για δομές επιχειρημάτων που έδιναν πάντα σωστά αποτελέσματα, όταν ξεκινούσαν από σωστές υποθέσεις. Το παραδοσιακό παράδειγμα ξεκινά με το επιχείρημα ότι «Ο Σωκράτης είναι άνθρωπος και όλοι οι άνθρωποι είναι θνητοί, επομένως ο Σωκράτης είναι θνητός». Θεωρήθηκε ότι αυτοί οι νόμοι της σκέψεις κυβερνούν τη λειτουργία της νόησης. Από τη μελέτη τους ξεκίνησε το πεδίο που ονομάζεται λογική [3], [12].

Η λογική ασχολείται με τη μελέτη των έγκυρων επιχειρημάτων, δηλαδή με ποιους τρόπους οδηγούμαστε αναγκαστικά σε κάποια συμπεράσματα, αν δεχτούμε κάποιες υποθέσεις. Η λογική, κατά τη συμβατική ερμηνεία της, απαιτεί γνώση του κόσμου που να είναι βέβαιη, συνθήκη που στην πραγματικότητα σπάνια επιτυγχάνεται. Για παράδειγμα, ενώ οι κανόνες του σκακιού ή της αριθμητικής είναι ορισμένοι, υπάρχουν καταστάσεις όπου η γνώση είναι περιορισμένη και όπου είναι αδύνατον να περιγράφονται με ακρίβεια τόσο η ίδια η κρατούσα κατάσταση, όσο και ένα ή περισσότερα μελλοντικά αποτελέσματα (π.χ. πολεμικές συγκρούσεις). Η θεωρία των πιθανοτήτων καλύπτει αυτό το κενό, επιτρέποντας την αυστηρή συλλογιστική υπό καθεστώς αβεβαιότητας και ρίσκου. Ουσιαστικά, επιτρέπει την κατασκευή ενός ολοκληρωμένου μοντέλου ορθολογικής σκέψης, το οποίο οδηγεί από ανεπεξέργαστες αντιληπτικές πληροφορίες στην κατανόηση του τρόπου με τον οποίο λειτουργεί ο κόσμος, και σε προβλέψεις για το μέλλον. Η κατασκευή αυτή ορίζεται ως ευφυής πράκτορας [3], [13].

Πράκτορας (agent) μπορεί να θεωρηθεί οτιδήποτε αντιλαμβάνεται το περιβάλλον του μέσα από αισθητήρες (sensors), αποτελεί μέρος του περιβάλλοντος αυτού, κάνει συλλογισμούς και δρα σε αυτό το περιβάλλον για να το αλλάξει μέσα από μηχανισμούς δράσης (effectors ή actuators) βάσει των στόχων για τους οποίους έχει σχεδιαστεί (Σχήμα 1.2). Ο βασικός αυτός ορισμός περιγράφει έναν άνθρωπο, ένα ρομπότ, ένα πρόγραμμα υπολογιστή ή και έναν απλό αισθητήρα. Εάν ο Πράκτορας (agent) είναι απλώς κάτι που «πράττει» (ο αγγλικός όρος agent επίσης προέρχεται από το λατινικό agere, που σημαίνει «πράττω»), οι αυτόνομοι πράκτορες ΤΝ πρέπει να κάνουν περισσότερα: να λειτουργούν αυτόνομα, να αντιλαμβάνονται το περιβάλλον τους, να διατηρούνται για ένα παρατεταμένο

χρονικό διάστημα, να προσαρμόζονται στις αλλαγές, και να δημιουργούν και να επιδιώκουν στόχους. [14].



Σχήμα 1.2: Βασικός Πράκτορας κατά Russell και Norvig [15]

Οι πράκτορες πρέπει να δρουν ορθολογικά. Ορθολογικός πράκτορας (rational agent) είναι ένας πράκτορας που ενεργεί έτσι ώστε να επιτυγχάνει το καλύτερο αποτέλεσμα ή, όταν υπάρχει αβεβαιότητα, το καλύτερο αναμενόμενο αποτέλεσμα. Οι ορθολογικοί πράκτορες λαμβάνουν αποφάσεις και δρουν βάσει μιας διαδικασίας λογικής απόδειξης. Στηρίζονται σε μία βάση γνώσης (knowledge base), χάρη στην οποία διατηρούν την αντίληψή τους για το περιβάλλον με μορφή λογικών προτάσεων (logic formulae), και σε ένα σύνολο κανόνων συμπερασμού, για να δράσουν. Έτσι, οι δράσεις που θα εκτελεστούν ανάγονται σε προβλήματα απόδειξης της μαθηματικής λογικής και πολλές φορές χρησιμοποιούνται συμπερασματικές μηχανές, για να επιτευχθούν. Οι ορθολογικοί πράκτορες εκτελούν συνεχώς τις εξής τρεις λειτουργίες:

- αντιλαμβάνονται τις δυναμικές συνθήκες του περιβάλλοντος,
- συλλογίζονται, ώστε να ερμηνεύσουν αυτά που αντιλαμβάνονται, να λύσουν προβλήματα, να συμπεράνουν και να καθορίσουν τη δράση τους,
- δρουν στο περιβάλλον, ώστε να το αλλάξουν.

Οι ορθολογικοί πράκτορες χρησιμοποιούν μεθόδους και τεχνικές που έχουν καθορισμένη και απλή σημασιολογία. Συνεπώς, είναι περισσότερο κατάλληλοι για στατικά περιβάλλοντα, ενώ καθίσταται ιδιαίτερα δύσκολη η εξαγωγή συμπερασμάτων σε μεταβαλλόμενα (δυναμικά) περιβάλλοντα, όπου απαιτείται επανασχεδιασμός των δράσεων του πράκτορα και μάλιστα σε περιορισμένα χρονικά όρια. Επίσης, η δράση ενός ορθολογικού πράκτορα σε ένα δυναμικό περιβάλλον απαιτεί ακριβή και ικανοποιητική (αν

όχι πλήρη) συμβολική περιγραφή του πραγματικού κόσμου, το οποίο είναι ανέφικτο σε πρακτικό επίπεδο [15].

Στην προσέγγιση της ΤΝ με τους Νόμους της Σκέψης, έμφαση δίνεται στη σωστή εξαγωγή συμπερασμάτων (συμπερασμός-inference), στην εξαγωγή δηλαδή συμπερασμάτων βάσει δοσμένων υποθέσεων. Ο συμπερασμός είναι συνήθως μια διαδικασία που περιλαμβάνει ένα ή περισσότερα βήματα. Σε αυτή την προσέγγιση ένα βασικό πρόβλημα αποτελεί η σχεδίαση ορθολογικών πρακτόρων, πρακτόρων που μπορούν να συμπεράνουν ότι μια δεδομένη ενέργεια είναι η βέλτιστη και έπειτα να ενεργούν με βάση αυτό το συμπέρασμα, καθώς υπάρχουν και περιπτώσεις όπου κάποιος ενεργεί ορθολογικά όχι όμως ως αποτέλεσμα της σωστής εξαγωγής συμπεράσματος αλλά αντανακλαστικά π.χ. η άμεση απομάκρυνση από μια εστία φωτιάς. Η προσέγγιση των ορθολογικών πρακτόρων στην ΤΝ έχει δύο πλεονεκτήματα έναντι των άλλων προσεγγίσεων: α) είναι πιο γενική από την προσέγγιση με τους «νόμους της σκέψης και β) προσφέρεται περισσότερο για επιστημονική ανάπτυξη [3].

Τέλος, τα επιτραπέζια παιχνίδια στρατηγικής αποτελούν σημαντικό στοιχείο έρευνας, πηγή έμπνευσης και δημιουργίας στο χώρο της τεχνητής νοημοσύνης (ΤΝ).

1.2 Συνεισφορά Διπλωματικής Εργασίας

Η ανάλυση επιτραπέζιων παιχνιδιών στρατηγικής με τη χρήση ευφυών πρακτόρων είναι αρκετά διαδομένη. Οι περισσότεροι αναλυτές βέβαια έχουν ασχοληθεί κυρίως με πολύ ανταγωνιστικά παιχνίδια, όπως για παράδειγμα το σκάκι, τα οποία είναι και τα πιο δημοφιλή στο ευρύ κοινό. Παρ' όλα αυτά, φιλότιμες προσπάθειες έχουν πραγματοποιηθεί και για το Mastermind, αν αναλογισθεί κανείς, πως έχει ασχοληθεί με αυτό και «ο πατέρας της ανάλυσης των αλγορίθμων» Donald Knuth. Στο πλαίσιο της εργασίας θα γίνει παρουσίαση και ανάλυση διαφόρων στρατηγικών επίλυσης του συγκεκριμένου παιχνιδιού.

Οι στρατηγικές αυτές δεν είναι κατ' ανάγκη καινούριες, ούτε άγνωστες. Για τις ανάγκες της εργασίας, έχουν δημιουργηθεί 6 διαφορετικοί αλγόριθμοι-στρατηγικές: Serial Algorithm, Simple Algorithm, Knuth Algorithm, Expected Size Algorithm, Max Entropy Algorithm, Serial With Worst Case Algorithm. Σημειώνεται ότι ο αλγόριθμος «Serial With Worst Case Algorithm», είναι μία νέα στρατηγική η οποία αναπτύχθηκε στο πλαίσιο της εργασίας. Οι παραπάνω στρατηγικές υιοθετήθηκαν αρχικά στην κλασσική εκδοχή του παιχνιδιού (6 διαθέσιμα χρώματα και κωδικός των 4 χρωμάτων). Στη συνέχεια, οι στρατηγικές δοκιμάσθηκαν και σε παραλλαγές του παιχνιδιού, οι οποίες σχεδιάστηκαν για τις ανάγκες τις παρούσας εργασίας, με αυξημένο όμως επίπεδο δυσκολίας, όπως:

- 8 διαθέσιμα χρώματα και κωδικός των 4 χρωμάτων
- 6 διαθέσιμα χρώματα και κωδικός των 5 χρωμάτων
- 7 διαθέσιμα χρώματα και κωδικός των 5 χρωμάτων

Τα αποτελέσματα ήταν αρκετά ικανοποιητικά, καθώς ο πιο αποδοτικός αλγόριθμος που σχεδιάστηκε μπορεί να εντοπίσει τον κρυφό κωδικό το πολύ σε 5 προσπάθειες (κλασσική έκδοση παιχνιδιού).

Τέλος, στο πλαίσιο της εργασίας αναπτύχθηκε και κατάλληλο γραφικό περιβάλλον, για την καλύτερη οπτικοποίηση των αποτελεσμάτων, χωρίς όμως να αποτελεί κύριο στόχο της εργασίας.

1.3 Δομή Διπλωματικής Εργασίας

Συνοπτικά, η εργασία δομείται στα παρακάτω κεφάλαια και ενότητες.

Στην Εισαγωγή παρουσιάστηκαν εισαγωγικές έννοιες σχετικές με την Τεχνητή Νοημοσύνη (ΤΝ) και προσδιορίστηκε το θέμα που θα εξετασθεί στο πλαίσιο της εργασίας (Σχεδίαση και Ανάπτυξη Αυτόνομου Πράκτορα για το Επιτραπέζιο Παιχνίδι Mastermind), καθώς και ο τρόπος επίλυσης προγραμματιστικά του συγκεκριμένου προβλήματος.

Στο δεύτερο κεφάλαιο γίνεται αναφορά στη σχέση μεταξύ επιτραπέζιων παιχνιδιών και Τεχνητής Νοημοσύνης όπως και στα Διασκεδαστικά Μαθηματικά (Recreational Mathematics). Εξετάζονται τα επιτεύγματα στα πιο διαδεδομένα παιχνίδια, όπως το σκάκι, και επισημαίνεται ο διαχωρισμός των παιχνιδιών, ανάλογα με το περιβάλλον και τον τρόπο παιξίματός τους.

Στο τρίτο κεφάλαιο, περιγράφεται το συγκεκριμένο παιχνίδι (Mastemind) που θα μελετηθεί στο πλαίσιο της εργασίας και περιγράφονται οι κανόνες διεξαγωγής του. Επιπρόσθετα, γίνεται αναφορά σε προγενέστερες μελέτες που έχουν πραγματοποιηθεί από άλλους αναλυτές.

Στο τέταρτο κεφάλαιο, αναλύονται όλοι οι αλγόριθμοι που έχουν υλοποιηθεί προγραμματιστικά με τη χρήση του περιβάλλοντος Java για την εύρεση του κρυφού κωδικού. Επίσης, παρατίθενται και αντίστοιχα παραδείγματα για την καλύτερη κατανόησή τους.

Στο πέμπτο κεφάλαιο, παρουσιάζονται τα αποτελέσματα των αλγορίθμων για όλες τις παραλλαγές του παιχνιδιού, όπως προκύπτουν από τον πράκτορα που έχει αναπτυχθεί. Ακολουθεί σχολιασμός των τεχνικών επίλυσης και σύγκριση μεταξύ τους.

Στο έκτο και τελευταίο κεφάλαιο, αποτυπώνονται τα συμπεράσματα που προκύπτουν από την προτεινόμενη προσέγγιση του προβλήματος καθώς και οι βελτιώσεις που μπορούν να πραγματοποιηθούν.

2 ΥΠΟΒΑΘΡΟ

2.1 Ιστορική Αναδρομή: Αυτόνομοι Πράκτορες και Παίγνια

Ο πρώτος ηλεκτρονικός ψηφιακός υπολογιστής γενικής χρήσης στον κόσμο ήταν ο ENIAC (Electronic Numerical Integrator and Computer), ο οποίος τέθηκε σε λειτουργία για πρακτικούς σκοπούς στα τέλη του 1945. Αρχικά το 1942 ο φυσικός John Mauchly πρότεινε μια εξ ολοκλήρου ηλεκτρονική μηχανή υπολογισμού. Εν τω μεταξύ, ο στρατός των ΗΠΑ, χρειάστηκε να υπολογίσει πολύπλοκους πίνακες βολής εν καιρώ πολέμου, και το αποτέλεσμα ήταν ο ENIAC, που κατασκευάστηκε μεταξύ 1943 και 1945 και ήταν ο πρώτος υπολογιστής μεγάλης κλίμακας που λειτουργούσε χωρίς να επιβραδύνεται από κανένα μηχανικό μέρος. Για μια δεκαετία ο ENIAC είχε πραγματοποιήσει περισσότερους υπολογισμούς από ό,τι είχε κάνει όλη η ανθρωπότητα μέχρι εκείνο το σημείο [16].

Η επανάσταση αυτή στον τομέα τον υπολογιστών δεν άργησε να δημιουργήσει και την ιδέα της ανάπτυξης της Τεχνητής Νοημοσύνης. Ήδη από τις αρχές της δεκαετίας του 1950 και πιο συγκεκριμένα το 1951 δημιουργήθηκε το πρώτο πρόγραμμα ΤΝ στο πανεπιστήμιο του Μάντσεστερ. Το πρόγραμμα αναπτύχθηκε για τον υπολογιστή Ferranti Mark 1 και ήταν προγραμματισμένο να μπορεί να παίζει αυτόνομα, σαν πράκτορας, τα δημοφιλή παιχνίδια σκάκι (Chess) και ντάμα (Checkers). Η ιδέα ενός αυτόνομου πράκτορα, ο οποίος θα μπορούσε να παίζει επιτραπέζια παιχνίδια είχε πλέον ολοκληρωθεί. Επόμενος στόχος ήταν ο πράκτορας να μπορεί να γίνει ακόμα πιο ανταγωνιστικός και να μπορεί να έρθει αντιμέτωπος με ανθρώπινους παίκτες. Την ίδια δεκαετία (1959) ο Arthur Samuel δημιούργησε το πρώτο πρόγραμμα που μπορούσε να μαθαίνει από μόνο του, προκειμένου να μπορεί να ανταγωνιστεί οποιοδήποτε παίκτη στην ντάμα. Ο Samuel όρισε τη «Μηχανική Μάθηση» ως ένα «πεδίο μελέτης που δίνει στους υπολογιστές τη δυνατότητα να μαθαίνουν χωρίς να είναι ρητά προγραμματισμένοι» [17], [18], [19].

Το 1990 αναπτύχθηκε η έννοια της ενισχυτικής μάθησης (reinforcement learning). Σε αυτήν την προσέγγιση το πρόγραμμα παίζει πολλά παιχνίδια εναντίον του εαυτού του και χρησιμοποιεί το σήμα «ανταμοιβής» (ή «τιμωρίας») στο τέλος κάθε παιχνιδιού για να βελτιώσει σταδιακά την ποιότητα των κινήσεων. Αναπτύχθηκε ένα πρόγραμμα για το παιχνίδι τάβλι (Backgammon), από την IBM και πιο συγκεκριμένα από τον Gerald Tesauro, το οποίο μπορούσε να αναμετρηθεί με επαγγελματίες παίκτες, αναδεικνύοντας με αυτόν τον τρόπο την αλματώδη εξέλιξη που είχε επέλθει στον τομέα της ΤΝ [20], [21]. Τέσσερα χρόνια

αργότερα το 1994, το υπολογιστικό πρόγραμμα Chinook, το οποίο ήταν προγραμματισμένο για να παίζει ντάμα, κατάφερε να κερδίσει τον δεύτερο καλύτερο παίκτη στον κόσμο Don Lafferty και να κατακτήσει το Εθνικό Πρωτάθλημα του παιχνιδιού στις ΗΠΑ [22].

Άλλη μια μεγάλη επιτυχία σημειώθηκε το 1997. Η εταιρεία IBM δημιούργησε την υπολογιστική μηχανή Deep Blue, έναν υπολογιστή, ο οποίος μπορούσε να παίζει σκάκι σε επαγγελματικό επίπεδο. Στις 11 Μαΐου 1997 ο παγκόσμιος πρωταθλητής Garry Kasparov θα ηττηθεί στον έκτο γύρο και το Deep Blue θα γίνει το πρώτο υπολογιστικό σύστημα το οποίο καταφέρνει να κερδίσει έναν παγκόσμιο πρωταθλητή σε έναν αγώνα, όπου εφαρμόζονται οι τυπικοί κανόνες χρόνου [23].

Μερικά χρόνια αργότερα ο υπερυπολογιστής της IBM Watson είχε «κατακτήσει» το παιχνίδι Jeopardy. Το Jeopardy ήταν μια αμερικανική εκπομπή παιχνιδιών, ένας διαγωνισμός κουίζ που αντιστρέφει την παραδοσιακή μορφή ερωτήσεων και απαντήσεων πολλών εκπομπών κουίζ. Αντί να δίνονται ερωτήσεις, στους διαγωνιζόμενους δίνονται ενδείξεις γενικής γνώσης με τη μορφή απαντήσεων και πρέπει να προσδιορίσουν το άτομο, το μέρος, το πράγμα ή την ιδέα που περιγράφει το στοιχείο, διατυπώνοντας κάθε απάντηση με τη μορφή ερώτησης. Ο υπερυπολογιστής αγωνίστηκε κόντρα σε δύο πρωταθλητές και κατάφερε να κερδίσει [24].

Την δική τους θέση έχουν και τα τυχερά παίγνια στην μελέτη των ειδικών. Ερευνητές από το Πανεπιστήμιο Carnegie Mellon (CMU) δημιούργησαν το σύστημα Τεχνητής Νοημοσύνης Libratus, Το Libratus ήρθε αντιμέτωπο με 4 πρωταθλητές πόκερ στο Texas Hold'em Poker (από τα πιο δημοφιλή τουρνουά πόκερ, διάρκειας 20 ημερών). Το κύριο χαρακτηριστικό του Libratus δεν είχε καμία προηγούμενη εκπαίδευση. Δεν δόθηκε στο Libratus κανένα δεδομένο πόκερ για να μάθει να παίζει. Του δόθηκαν απλώς οι κανόνες παιχνιδιού κι εκείνο αποφάσιζε τη στρατηγική του. Ο αλγόριθμός έμαθε μόνος του να παίζει πολύ καλά ξεκινώντας από το μηδέν [25].

Τα παιχνίδια (επιτραπέζια, παιχνίδια με τράπουλες) είναι ένας δημοφιλής τομέας για έρευνα στο πεδίο της ΤΝ, καθώς μπορούν να χρησιμοποιηθούν ως πλατφόρμες για την ανάπτυξη νέων εφαρμογών ΤΝ, αλλά και δίνουν τη δυνατότητα μέτρησης του πόσο καλά λειτουργούν. Επιπλέον, τα παιχνίδια μπορούν να αποδείξουν ότι οι μηχανές είναι ικανές να συμπεριφέρονται με ευφυΐα, χωρίς να θέτουν σε κίνδυνο ανθρώπινες ζωές ή περιουσίες [26].

2.2 Κατηγοριοποίηση παιχνιδιών

Οι ερευνητές της ΤΝ, μέσω της μελέτης των παιχνιδιών, προσπάθησαν να μελετήσουν προβλήματα αναζήτησης με αντιπαλότητα, καθώς η κατάσταση ενός παιχνιδιού είναι εύκολο να αναπαρασταθεί και οι πράκτορες περιορίζονται συνήθως σε ένα μικρό αριθμό ενεργειών που τα αποτελέσματά του ορίζονται από ακριβείς κανόνες. Τα προβλήματα αυτά αναφέρονται ως ανταγωνιστικά παίγνια ή παίγνια δύο αντιπάλων (adversary ή two-persons games) και επιλύονται με ειδικούς αλγόριθμους αναζήτησης [3], [4].

Τα παιχνίδια μπορούν να διαχωρισθούν με βάση την πληροφορία που παρέχεται στους παίχτες. Τα παιχνίδια όπου όλες οι πληροφορίες που αφορούν τους παίκτες είναι γνωστές καλούνται παίγνια τέλειας πληροφόρησης. Η τέλεια πληροφόρηση (perfect information), είναι βασική έννοια στη θεωρία παιγνίων[1]. Η τέλεια πληροφόρηση αναφέρεται στο γεγονός ότι όλοι οι παίκτες έχουν ακριβώς την ίδια πληροφορία, σχετικά με το παιχνίδι, γνωρίζουν όλα τα στοιχεία του παιχνιδιού και η πληροφορία αυτή είναι κοινή γνώση. Από τα πιο γνωστά παιχνίδια τέλειας πληροφόρησης είναι το σκάκι και το Go. Η τέλεια πληροφόρηση σημαίνει ότι κάθε φορά μόνο ένας από τους παίκτες κινείται, ότι το παιχνίδι εξαρτάται μόνο από τις επιλογές τους, θυμούνται το παρελθόν και γνωρίζουν όλες τις πιθανές μελλοντικές καταστάσεις του παιχνιδιού [27]. Για παράδειγμα στο σκάκι, το ταμπλό και τα πιόνια είναι ευδιάκριτα και στους δύο συμμετέχοντες καθ' όλη την διάρκεια του παιχνιδιού.

Στον αντίποδα, η μη τέλεια πληροφόρηση (imperfect information), αφορά καταστάσεις όπου ο ένας παίκτης δεν γνωρίζει (ή δεν θυμάται) τις κινήσεις του αντιπάλου και ως αποτέλεσμα χρειάζεται να αξιολογηθούν, όλα τα πιθανά αποτελέσματα, προτού ληφθεί μία απόφαση [28], [29]. Το πόκερ είναι ένα τέτοιο παράδειγμα, καθώς δεν είναι ορατές όλες οι κάρτες σε όλους τους παίχτες.

Τα παιχνίδια επίσης χωρίζονται σε αιτιοκρατικά και σε παιχνίδια που περιλαμβάνουν ένα στοιχείο τύχης (π.χ. ρίψη ζαριών). Αιτιοκρατικό ή ντετερμινιστικό, ονομάζεται ένα παιχνίδι όταν οι κινήσεις του παίκτη δεν επηρεάζονται από τον παράγοντα τύχη, αλλά προέρχονται από εντελώς προβλέψιμα αποτελέσματα. Για παράδειγμα, το σκάκι, όπου οι κανόνες δεν επιτρέπουν καμία παραλλαγή του αποτελέσματος και δεν εμπλέκονται φυσικοί παράγοντες, είναι ένα αιτιοκρατικό παιχνίδι. Στα ντετερμινιστικά παιχνίδια, μπορούν να ενταχθούν τόσο αυτά που παρέχουν τέλεια πληροφόρηση, όσο και αυτά που δεν παρέχουν,

[1] Ο όρος «παίγνιο» αφορά την περιγραφή του τρόπου με τον οποίο παίζεται το παιχνίδι, και περιλαμβάνει τα αντικείμενα καθώς και το σύνολο των κανόνων.

π.χ. Ναυμαχία, όπου οι παίκτες δεν γνωρίζουν που έχουν τοποθετηθεί τα πλοία του αντιπάλου πάνω στο ταμπλό. Στα αιτιοκρατικά μερικώς παρατηρήσιμα παιχνίδια, η αβεβαιότητα οφείλεται αποκλειστικά στην έλλειψη πρόσβασης στις επιλογές του αντιπάλου [3].

Τα παιχνίδια που περιλαμβάνουν ένα στοιχείο τύχης καλούνται στοχαστικά παιχνίδια, Τα στοχαστικά παιχνίδια είναι λίγο πιο κοντά στο απρόβλεπτο της πραγματικής ζωής, ενδεικτικά η ρίψη ζαριών. Το τάβλι είναι ένα αντιπροσωπευτικό στοχαστικό παιχνίδι που συνδυάζει και τύχη και ικανότητα. Κάθε στιγμή, τόσο το ταμπλό, όσο και οι διαθέσιμες κινήσεις είναι ορατές και στους δύο παίκτες. Δεν γνωρίζουν όμως τα αποτελέσματα από τις ρίψεις ζαριών που θα ακολουθήσουν. Αυτό σημαίνει ότι δεν μπορούν να γνωρίζουν τις νόμιμες κινήσεις του αντιπάλου και ως αποτέλεσμα να μην μπορεί να δημιουργηθεί ένα κανονικό δένδρο παιχνιδιού, όπως στο σκάκι [3].

2.3 Προβλήματα Ικανοποίησης Περιορισμών (CSP)

Υπάρχουν διάφορες τεχνικές για την επίλυση προβλημάτων. Μία τέτοια τεχνική αποτελεί η ικανοποίηση περιορισμών και τα προβλήματα που μελετάει είναι γνωστά ως Προβλήματα Ικανοποίησης Περιορισμών (CSP - Constraint Satisfaction Problems). Η ικανοποίηση περιορισμών είναι μια τεχνική όπου ένα πρόβλημα επιλύεται, όταν οι τιμές των μεταβλητών του ικανοποιούν ορισμένους περιορισμούς ή κανόνες του προβλήματος. Η συγκεκριμένη τεχνική οδηγεί σε βαθύτερη κατανόηση της δομής του προβλήματος, καθώς και της πολυπλοκότητάς του. Ένα CSP αποτελείται από τρεις συνιστώσες:

- X: Ένα σύνολο μεταβλητών, $\{X_1,\ldots X_n\}$.
- D: Ένα σύνολο πεδίων τιμών, $\{D_1,\ldots D_n\}$, ένα για κάθε μεταβλητή.
- C: Ένα σύνολο περιορισμών που καθορίζουν τους επιτρεπτούς συνδυασμούς τιμών.

Ένα πεδίο τιμών D_i, αποτελείται από ένα σύνολο επιτρεπτών τιμών, $\{v_1,\ldots,v_k\}$, για τη μεταβλητή X_i. Για παράδειγμα, μία λογική μεταβλητή (Boole) θα έχει το πεδίο τιμών {αληθές, ψευδές}. Διαφορετικές μεταβλητές μπορεί να έχουν διαφορετικά πεδία τιμών διαφορετικού μεγέθους. Η τιμή του περιορισμού αποτελείται από ένα ζεύγος <scope, rel>. Το scope (εμβέλεια) είναι μία πλειάδα μεταβλητών που συμμετέχουν στον περιορισμό και

rel είναι μία σχέση (relation) που περιλαμβάνει μία λίστα τιμών που μπορούν να πάρουν οι μεταβλητές για να ικανοποιήσουν τους περιορισμούς του προβλήματος.

Οι απαιτήσεις για την επίλυση ενός CSP είναι να δημιουργηθεί ένας χώρος καταστάσεων και να κατανοηθεί πλήρως η κεντρική ιδέα του προβλήματος. Μία κατάσταση στο χώρο καταστάσεων ορίζεται με την ανάθεση τιμών σε ορισμένες ή όλες τις μεταβλητές, όπως π.χ. {X1 = ν1, X2 = ν2, …}. Μία ανάθεση που δεν παραβιάζει κανέναν περιορισμό ή κανόνα ονομάζεται συνεπής ή νόμιμη ανάθεση. Πλήρης είναι μία ανάθεση στην οποία κάθε μεταβλητή λαμβάνει μία τιμή και η λύση του CSP παραμένει συνεπής. Ενώ, μία ανάθεση που αναθέτει τιμές σε ορισμένες μόνο από τις μεταβλητές ονομάζεται μερική.

Τα CSP χαρακτηρίζονται από μεγάλο αριθμό μεταβλητών και πιθανών τιμών, γεγονός που οδηγεί στο φαινόμενο της συνδυαστικής έκρηξης, δηλαδή την πολύ μεγάλη αύξηση του χώρου αναζήτησης, κάνοντας την επίλυσή τους εξαιρετικά χρονοβόρα. Προβλήματα στα οποία είναι γνωστές μόνο κάποιες ιδιότητες των τελικών καταστάσεων και επιδιώκεται η εύρεση ενός πλήρους στιγμιότυπου τελικής κατάστασης είναι, CSPs όπως για παράδειγμα, τα προβλήματα χρονοπρογραμματισμού εργασιών. Επίσης, ως CSP μπορεί να αντιμετωπισθεί και το Sudoku, όπου ο βασικός περιορισμός είναι ότι κανένας αριθμός από το 0-9 δεν μπορεί να επαναληφθεί στην ίδια γραμμή ή στήλη. Άλλα παραδείγματα εφαρμογών αποτελούν ο χρωματισμός ενός χάρτη, η κατάστρωση timetable, ο γραμμικός προγραμματισμός, χρηματιστηριακή ανάλυση, κλπ. [3], [4].

3 ΠΕΡΙΓΡΑΦΗ ΠΑΙΧΝΙΔΙΟΥ MASTERMIND

3.1 Ιστορικό Παιχνιδιού

Το Mastermind είναι ένα επιτραπέζιο παιχνίδι για 2 παίκτες. Ο ένας παίκτης (κωδικοποιητής) δημιουργεί ένα χρωματικό κωδικό και ο άλλος παίκτης (αποκωδικοποιητής) επιχειρεί να τον εντοπίσει. Πρόγονος του παιχνιδιού θεωρείται το παιχνίδι Bulls and Cows, το οποίο απαιτούσε μόνο μολύβι και χαρτί. Τη δεκαετία του 1960, σχεδιάστηκε και υλοποιήθηκε στο Πανεπιστήμιο Cambridge, από τον Frank King, μια παραλλαγή του παιχνιδιού, το «MOO», με χρήση του υπολογιστικού συστήματος Titan. Είχε δημιουργηθεί επίσης άλλη μια εκδοχή για το σύστημα κοινής χρήσης χρόνου TSS/8, από τον J.S. Felton, ενώ τέλος ο Jerold Grochow δημιούργησε μία νέα εκδοχή για το σύστημα Multics στο MIT.

Η σύγχρονη εκδοχή του παιχνιδιού με καρφάκια επινοήθηκε το 1970 από τον Mordecai Meirowitz, έναν Ισραηλινό ταχυδρόμο και ειδικό στις τηλεπικοινωνίες. Ο Meirowitz παρουσίασε την ιδέα σε πολλές μεγάλες εταιρείες παιχνιδιών, αλλά μετά την επίδειξή του στην Διεθνή Έκθεση Παιχνιδιών που πραγματοποιήθηκε στην Νυρεμβέργη, η εταιρεία πλαστικών Invicta Plastics, από το Ηνωμένο Βασίλειο, αγόρασε όλα τα δικαιώματα του παιχνιδιού. Με την σειρά της η Invicta Plastics έδωσε άδεια κατασκευής παγκοσμίως στην Hasbro, με εξαίρεση την Pressman Toys και την Orda Industries που έχουν τα δικαιώματα κατασκευής στις Ηνωμένες Πολιτείες και το Ισραήλ, αντίστοιχα.

Το Mastermind πλέον έχει διάφορες παραλλαγές στον τρόπο παιχνιδιού του. Η αλλαγή του αριθμού των χρωμάτων και του μεγέθους του κωδικού, έχει σαν αποτέλεσμα να δημιουργείται ένα φάσμα παιχνιδιών διαφορετικών επιπέδων δυσκολίας. Μία άλλη κοινή παραλλαγή είναι η υποστήριξη διαφορετικών αριθμών παικτών που αναλαμβάνουν τους ρόλους του παίκτη κωδικοποιητή και του παίκτη αποκωδικοποιητή.

3.2 Διεξαγωγή Παιχνιδιού

Το παιχνίδι στην κλασσική του εκδοχή, δηλαδή σε αυτήν που συμμετέχουν 2 παίκτες, τα διαθέσιμα χρώματα είναι 6 και ο κωδικός έχει μέγεθος 4 χρωμάτων, παίζεται χρησιμοποιώντας τα παρακάτω (Σχήμα 3.1):

- Μια πλακέτα αποκωδικοποίησης, η οποία διαθέτει ειδική θέση για να τοποθετηθεί ο κρυφός κωδικός, 12 σειρές όπου ο παίκτης-λύτης τοποθετεί τις δοκιμές του, ενώ σε κάθε σειρά υπάρχουν στην άκρη 4 μικρές οπές, όπου τοποθετούνται τα αποτελέσματα του κάθε γύρου.
- Μεγάλα καρφάκια έξι διαφορετικών χρωμάτων, με την βοήθεια των οποίων σχηματίζεται ο κωδικός.
- Μικρά καρφάκια σε κόκκινο και λευκό χρώμα, τα οποία αξιοποιούνται για την πληροφορία του εκάστοτε αποτελέσματος.

Σχήμα 3.1: Το παιχνίδι Mastermind

3.3 Κανόνες Παιχνιδιού

Οι δύο παίκτες αποφασίζουν εκ των προτέρων πόσα παιχνίδια θα παίξουν, τα οποία πρέπει να είναι ζυγός αριθμός, προκειμένου να μπορούν να εναλλάσσουν τους ρόλους του κωδικοποιητή και του αποκωδικοποιητή. Επιπρόσθετα αποφασίζουν, εάν επιτρέπεται η χρήση διπλότυπων χρωμάτων. Εάν ναι, ο δημιουργός κωδικών μπορεί να επιλέξει ακόμη και τέσσερα καρφάκια κώδικα ίδιου χρώματος. Ο κωδικοποιητής επιλέγει ένα μοτίβο τεσσάρων δεσμών κώδικα και τοποθετεί το επιλεγμένο μοτίβο στην ειδική θέση που καλύπτεται από την ασπίδα, ορατή στον κωδικοποιητή, αλλά όχι στον αποκωδικοποιητή.

Ο αποκωδικοποιητής προσπαθεί να μαντέψει τον κωδικό, τόσο ως προς τη διάταξη, όσο και ως προς το χρώμα, μέσα σε δώδεκα προσπάθειες. Κάθε δοκιμή πραγματοποιείται τοποθετώντας μια σειρά από καρφάκια κώδικα στον πίνακα αποκωδικοποίησης. Μόλις τοποθετηθεί, ο κωδικοποιητής παρέχει ανατροφοδότηση τοποθετώντας από μηδέν έως τέσσερα μικρά καρφάκια πληροφορίας στις μικρές τρύπες της σειράς με την δοκιμή. Ένα μικρό κόκκινο καρφάκι πληροφορίας τοποθετείται για κάθε μεγάλο καρφάκι από την δοκιμή που είναι σωστό τόσο σε χρώμα όσο και σε θέση, ενώ εάν το χρώμα υπάρχει αλλά βρίσκεται σε λάθος θέση, τότε τοποθετείται ένα λευκό μικρό καρφάκι πληροφορίας.

Σε περίπτωση που υπάρχουν διπλότυπα χρώματα στην δοκιμή, το κάθε καρφάκι πληροφορίας, αντιστοιχεί σε ένα από τα καρφάκια στην δοκιμή. Για παράδειγμα, εάν ο κρυφός κωδικός είναι «Κόκκινο, Κόκκινο, Μπλε, Μπλε» και ο παίκτης μαντέψει «Κόκκινο, Κόκκινο, Κόκκινο, Μπλε», ο δημιουργός κωδικών θα απονείμει δύο κόκκινα καρφάκια πληροφορίας για τα δύο σωστά κόκκινα, τίποτα για το τρίτο κόκκινο, καθώς δεν υπάρχει ένα τρίτο κόκκινο στον κωδικό και ένα ακόμα κόκκινο καρφάκι πληροφορίας για το μπλε. Δεν δίνεται καμία ένδειξη για το γεγονός ότι ο κωδικός περιλαμβάνει και δεύτερο μπλε.

Μόλις παρέχονται σχόλια (ανατροφοδότηση), ξεκινάει μια νέα δοκιμή. Οι δοκιμές και τα σχόλια συνεχίζουν να εναλλάσσονται, έως ότου είτε ο αποκωδικοποιητής μαντέψει σωστά, είτε όλες οι σειρές στον πίνακα αποκωδικοποίησης γεμίσουν, γεγονός που υποδηλώνει και την νίκη του κωδικοποιητή.

3.4 Σχετικές Εργασίες

Το Mastermind αποτελεί ένα παιχνίδι δύο παικτών μηδενικού αθροίσματος (2-player 0-sum game). Αυτού του είδους τα παίγνια είναι ένα από τα βασικότερα μοντέλα στην θεωρία παιγνίων. Υπάρχουν δύο παίκτες, όπου ο καθένας διαθέτει ένα σύνολο στρατηγικών. Ενώ ο ένας παίκτης προσπαθεί να μεγιστοποιήσει το κέρδος του, ο αντίπαλος παίκτης προσπαθεί να ενεργήσει με τέτοιο τρόπο, ώστε να ελαχιστοποιήσει αυτήν την ανταμοιβή. Πιο απλά, το κέρδος ενός παίκτη συνεπάγεται την απώλεια του αντιπάλου παίκτη. Υπάρχουν δύο γενικοί τύποι παιχνιδιών μηδενικού αθροίσματος: αυτά με τέλεια πληροφόρηση και αυτά χωρίς [30].

Το 2005 ο Barteld Pieter Kooi, καθηγητής στο πανεπιστήμιο Groningen, αξιοποιώντας την παραπάνω θεώρηση και το γεγονός ότι το Mastermind αποτελεί παιχνίδι ατελούς πληροφόρησης (imperfect information), προσπάθησε να δημιουργήσει έναν αλγόριθμο επίλυσης του παιχνιδιού. Απέδειξε ότι εάν μπορέσουμε να διαχωρίσουμε το πλήθος όλων των πιθανών κωδικών *A*, σε περισσότερα σετ μικρότερου μεγέθους, και γνωρίζουμε σε πιο από όλα αυτά τα σετ ανήκει ο κωδικός που ψάχνουμε να εντοπίσουμε, τότε αυξάνεται και το αναμενόμενο κέρδος. Εάν, για παράδειγμα, ψάχνουμε να εντοπίσουμε μία συγκεκριμένη κάρτα από μία τράπουλα με 52 κάρτες, τότε το αναμενόμενο κέρδος είναι $\frac{1}{52}$. Εάν όμως χωρίσουμε αυτήν την τράπουλα σε δύο στοίβες με x και y κάρτες η κάθε μία, τότε το αναμενόμενο κέρδος διπλασιάζεται, καθώς εάν γνωρίζουμε σε ποια από τις δύο στοίβες υπάρχει η ζητούμενη κάρτα, τότε οι πιθανότητες διαμορφώνονται σε $\frac{1}{x}$ και $\frac{1}{y}$ αντίστοιχα. Έτσι,

$$\frac{x}{52}\cdot\frac{1}{x}+\frac{y}{52}\cdot\frac{1}{y}=\frac{2}{52}$$

Τα μεγέθη της κάθε στοίβας, δηλαδή τα μεγέθη x και y, είναι ανεξάρτητα και δεν επηρεάζουν την εν λόγω στρατηγική. Συνεπώς, εάν κάποιος θέλει να μεγιστοποιήσει τον αριθμό των συνδυασμών για τους οποίους θα κέρδιζε, θα πρέπει να μεγιστοποιήσει τον αριθμό των τμημάτων, στα οποία χωρίζεται το σύνολο όλων των συνδυασμών του προηγούμενου γύρου. Η προσέγγιση του Kooi είναι αρκετά αποτελεσματική, καθώς κατά μέσο όρο, επιτυγχάνει να επιλύει το Mastermind σε 4,373 δοκιμές [31].

Το 2007 οι Albrecht Heffer και Harold Heefer προσπάθησαν να βρουν μια διαφορετική λύση για μια παραλλαγή του παιχνιδιού το Logik. Το Logik είναι αρκετά όμοιο με το

Mastermind, καθώς έχουν τους ίδιους κανόνες. Οι μόνες διαφορές μεταξύ των δύο επιτραπέζιων παιχνιδιών είναι, ότι στο Logik οι δύο παίκτες παίζουν ταυτόχρονα, με τον νικητή να ανακηρύσσεται ο παίκτης που θα προλάβει πρώτος να βρει τον κωδικό. Επιπλέον ο κωδικός είναι μεγαλύτερου μεγέθους, κωδικός 5 χρωμάτων, με τα διαθέσιμα χρώματα όπως είναι λογικό να αυξάνονται και να γίνονται 8. Έτσι πλέον οι πιθανοί συνδυασμοί γίνονται $8^5 = 32768$ σε σχέση με το Mastermind που είναι $6^4 = 1296$. Οι Heffer και Heffer εντόπισαν στην τυπική απόκλιση ένα δείκτη της στατιστικής διασποράς των λύσεων στα αποτελέσματα που εξάγονται σε κάθε δοκιμή. Εάν x_i ο αριθμός των λύσεων για n πιθανά αποτελέσματα, ο κωδικός με την μικρότερη τυπική απόκλιση ρ, θα δώσει και την καλύτερη κατανομή [32].

$$\rho = \sqrt{\frac{n\sum_{i=1}^{n} x^2{}_i - \sum_{i=1}^{n}(x_i)^2}{n(n-1)}}$$

Ο Justin Dowell, στην εργασία του, εμπνευσμένος από τον Barteld Kooi, προσπαθεί να αξιοποιήσει την μελέτη του, έτσι ώστε να μπορέσει να φτιάξει και αυτός τον δικό του αλγόριθμο. Έτσι καταφέρνει και δημιουργεί δύο νέες στρατηγικές. Στην πρώτη, επιχειρεί να συνδυάσει τις στρατηγικές των Kooi και Heffer και Heefer, με σκοπό να δημιουργήσει έναν νέο, πιο αποτελεσματικό αλγόριθμο. Έτσι κάθε φορά επιλέγει τον κωδικό εκείνο που μεγιστοποιεί τον αριθμό των διαχωρισμών και ελαχιστοποιεί την τυπική απόκλιση αυτών των διαχωρισμών. Στην δεύτερη στρατηγική αξιοποιεί και πάλι την λογική του διαχωρισμού, όμως αυτή την φορά, θα επιλέγει τον κωδικό με τις περισσότερες κατατμήσεις και τη μέγιστη μείωση που θα επιφέρει στο σετ των διαθέσιμων κωδικών [33], [34].

Στον τομέα της Τεχνητής Νοημοσύνης άλλος ένας συνήθης τρόπος επίλυσης προβλημάτων είναι η δημιουργία και χρήση δικτύων Bayes. Η σύνταξη ενός δικτύου Bayes αποτελείται από έναν προσανατολισμένο άκυκλο γράφο (DAG) με ορισμένες τοπικές πιθανοτικές πληροφορίες που επισυνάπτονται σε κάθε κόμβο. Η σημασιολογία ορίζει τον τρόπο με τον οποίο η σύνταξη αντιστοιχεί σε μία συνδυασμένη κατανομή πιθανότητας για τις μεταβολές του δικτύου. Την τεχνική αυτή προσπάθησε να εκμεταλλευτεί ο Jiri Vomlel, προκειμένου να κατασκευάσει ένα δίκτυο Bayes, που να μπορεί να είναι ανταγωνιστικό στο επιτραπέζιο παιχνίδι Mastermind, κάνοντας χρήση των πιθανοτήτων [35].

Τέλος, το Mastermind μπορεί να θεωρηθεί ως ένα πρόβλημα ικανοποίησης περιορισμών (CSP), όπου οι θέσεις του χρωματικού κωδικού αντιστοιχούν στις μεταβλητές και τα διαθέσιμα χρώματα στις τιμές που μπορούν να πάρουν. Ωστόσο, υπάρχουν και

κάποιες σημαντικές διαφορές που το διαφοροποιούν σε σχέση με τα κλασσικά CSPs. Πρώτον, δεν είναι γνωστοί εξ αρχής όλοι οι περιορισμοί, αλλά αποκαλύπτονται σταδιακά μέσα από τις δοκιμές του παίκτη-αποκωδικοποιητή. Δεύτερον, η ανατροφοδότηση του κωδικοποιητή έχει πάντα ως αναφορά έναν πλήρη χρωματικό κωδικό που δίνεται ως δοκιμή, καθώς και τον κρυφό κωδικό, οπότε οι περιορισμοί εκφράζονται με έναν σχετικό (ως προς την αντίστοιχη δοκιμή και τον κρυφό κωδικό) και όχι απόλυτο τρόπο. Παρόλο που κάποιες ιδέες από τον χώρο των CSPs θα μπορούσαν να φανούν χρήσιμες και στο Mastermind, οι διαφορές που αναφέρθηκαν δεν ευνοούν την χρήση γενικών, off-the-shelf CSP τεχνικών, όπως φαίνεται και από την έλλειψη σχετικών εργασιών στη βιβλιογραφία.

4 Η ΔΙΚΗ ΜΑΣ ΠΡΟΣΕΓΓΙΣΗ

Στο παρόν κεφάλαιο παρουσιάζεται ο τρόπος με τον οποίο αντιμετωπίστηκε το πρόβλημα της δημιουργίας ενός αυτόνομου πράκτορα για το επιτραπέζιο παιχνίδι Mastermind. Παρουσιάζονται αναλυτικά τόσο η λογική που ακολουθήθηκε, όσο και ο τρόπος που αυτή υλοποιήθηκε.

4.1 Υλοποίηση

Το παιχνίδι είναι προγραμματισμένο σε γλώσσα Java και για την χρήση των γραφικών έχει αξιοποιηθεί η πλατφόρμα ανοικτού κώδικα JavaFX, η οποία παρέχει ποικίλες δυνατότητες για την δημιουργία γραφικών περιβαλλόντων.

Ποιο συγκεκριμένα για τον πράκτορα έχουν υλοποιηθεί 14 κλάσεις.

- **Algorithm** Class: Περιέχει βασικές μεθόδους που χρησιμοποιούν οι αλγόριθμοι Simple, Maximum Entropy, Expected Size, Knuth.
- **AlgorithmWithThread** Class: Περιέχει βασικές μεθόδους που χρησιμοποιούν οι αλγόριθμοι Maximum Entropy, Expected Size, Knuth όταν αξιοποιούν πολυνηματικό προγραμματισμό.
- **EntropyAlgorithm** Class: Περιλαμβάνει την λογική του αντίστοιχου αλγορίθμου.
- **EntropyThread** Class: Περιλαμβάνει την λογική του αλγορίθμου όταν γίνεται χρήση πολυνηματικού προγραμματισμού.
- **CalculateEntropy** Class: Υπολογίζει την εντροπία στον πολυνηματικό προγραμματισμό.
- **ExpectedSizeAlgorithm** Class: Περιλαμβάνει την λογική του αντίστοιχου αλγορίθμου.
- **ExpectedSizeThread** Class: Περιλαμβάνει την λογική του αλγορίθμου όταν γίνεται χρήση πολυνηματικού προγραμματισμού.
- **CalculateSize** Class: Υπολογίζει το αναμενόμενο μέγεθος στον πολυνηματικό προγραμματισμό.
- **KnuthAlgorithm** Class: Περιλαμβάνει την λογική του αντίστοιχου αλγορίθμου.
- **KnuthThread** Class: Περιλαμβάνει την λογική του αλγορίθμου όταν γίνεται χρήση πολυνηματικού προγραμματισμού.

- **CalculateWorstCase** Class: Υπολογίζει την ελάχιστη χείριστη κατάσταση στον πολυνηματικό προγραμματισμό.
- **SimpleAlgorithm** Class: Περιλαμβάνει την λογική του αντίστοιχου αλγορίθμου.
- **SerialAlgorithm** Class: Περιλαμβάνει την λογική του αντίστοιχου αλγορίθμου.
- **SerialAlgorithmWorstCase** Class: Περιλαμβάνει την λογική του αντίστοιχου αλγορίθμου.

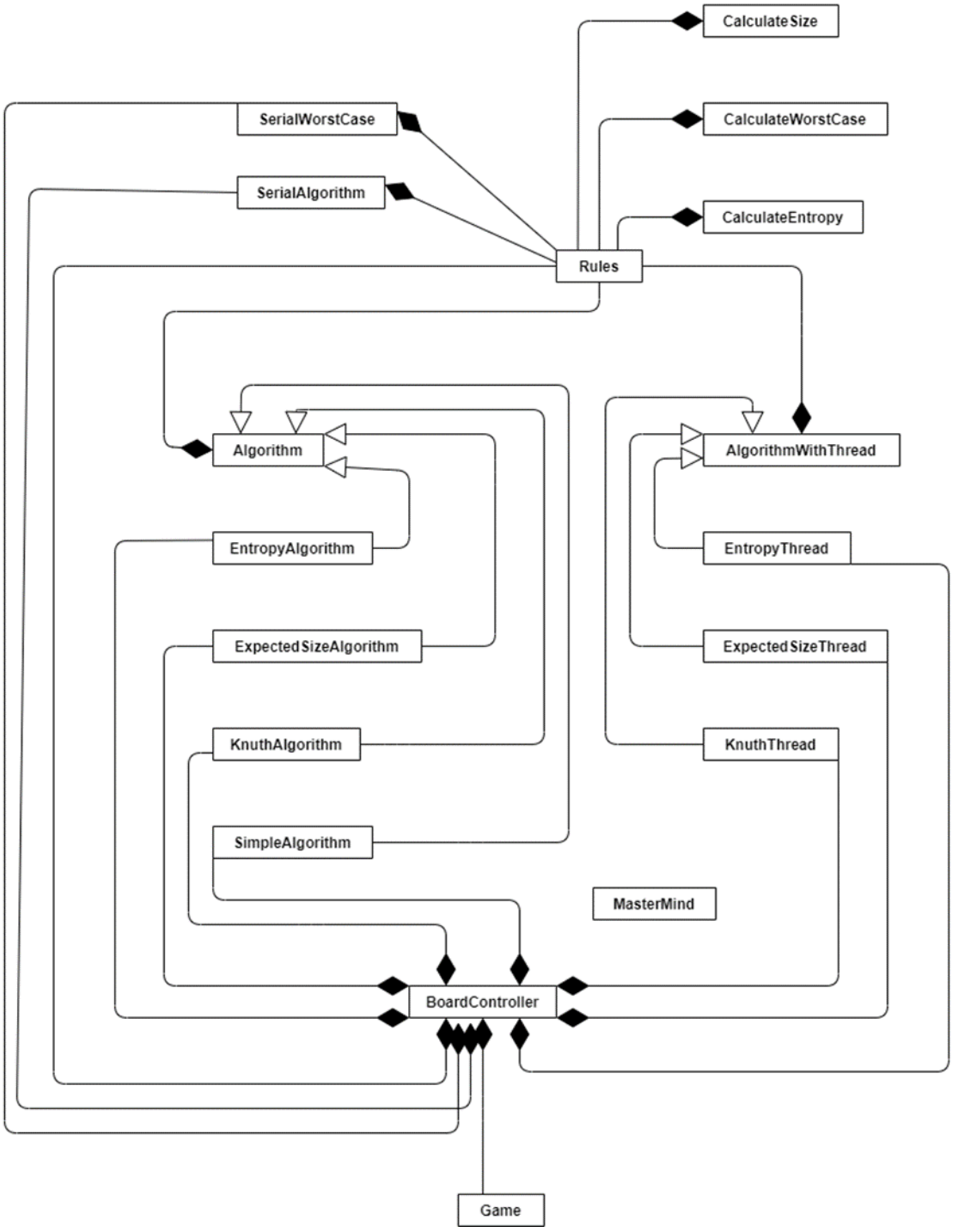
Για την λογική του παιχνιδιού έχουν δημιουργηθεί 2 κλάσεις:

- **Game** class: Περιέχει τις απαραίτητες αρχικοποιήσεις, όπως για παράδειγμα την ανάθεση τιμών στα χρώματα και τη δημιουργία του κρυφού κωδικού.
- **Rules** class: Περιέχει τους κανόνες του παιχνιδιού, την συνάρτηση που εξάγει το αποτέλεσμα της δοκιμής με τον κρυφό κωδικό και την συνάρτηση που ελέγχει εάν το παιχνίδι έχει ολοκληρωθεί.

Και τέλος για το γραφικό περιβάλλον που βλέπει ο χρήστης έχουν δημιουργηθεί τα εξής:

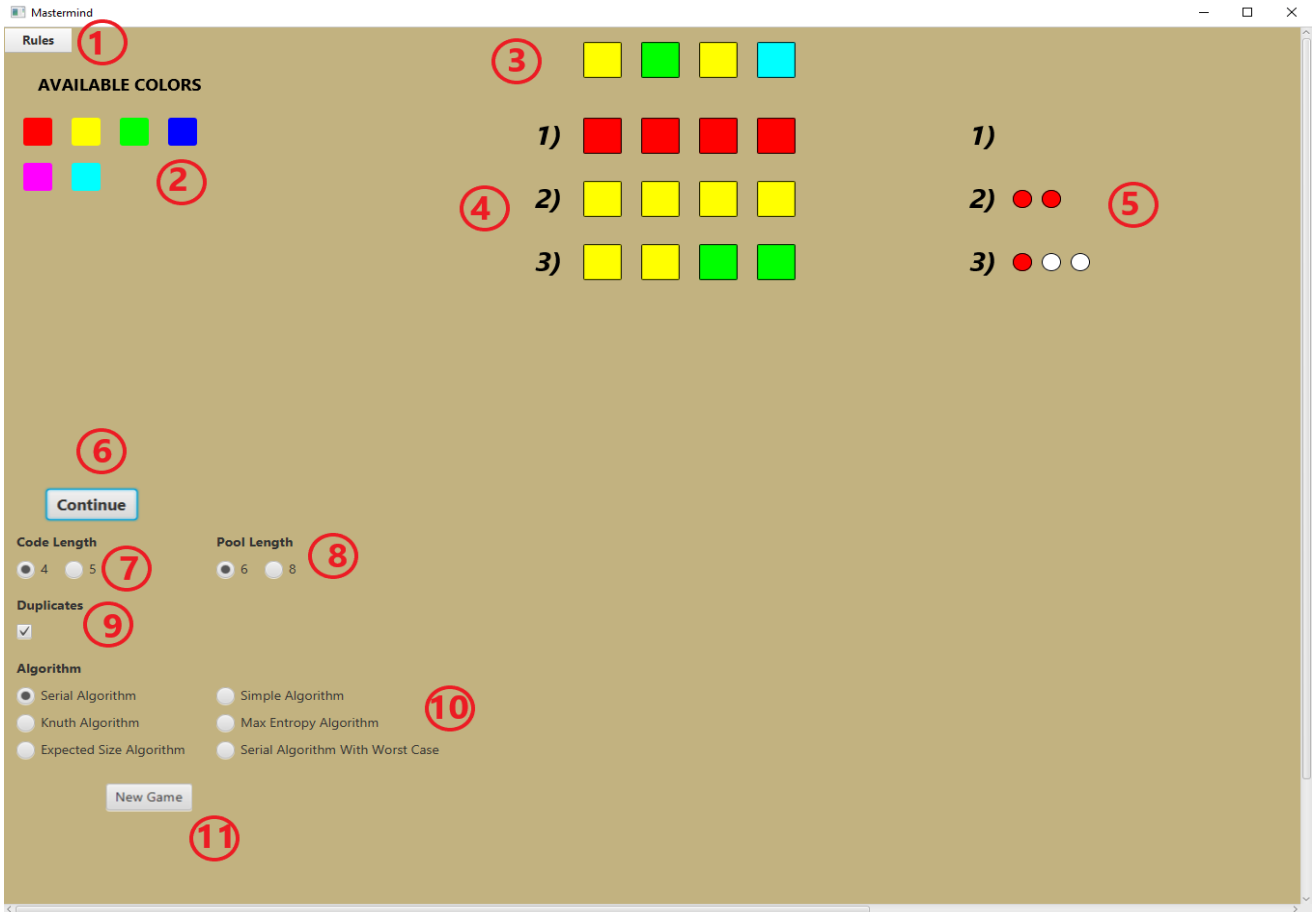
- **Board.fxml**: Είναι το αρχείο που συνδέεται με την πλατφόρμα JavaFX και δημιουργεί το γραφικό περιβάλλον.
- **BoardController** Class: Περιέχει όλες τις συναρτήσεις που συνδέουν το γραφικό περιβάλλον με τον πράκτορα.
- **MasterMind** Class: Είναι η κύρια κλάση του προγράμματος. Ενώνει όλες τις υπόλοιπες κλάσεις μεταξύ τους και περιέχει την main συνάρτηση από την οποία «τρέχει» όλο το πρόγραμμα.

Στην Εικόνα 4.1 παρουσιάζεται το Class Diagram της υλοποίησης.



Εικόνα 4.1: Class Diagram

Στην Εικόνα 4.2 παρουσιάζεται το γραφικό περιβάλλον, που εμφανίζεται στον χρήστη,



Εικόνα 4.2: Γραφικό περιβάλλον εφαρμογής

1) Από το μενού «Rules» ο παίκτης μπορεί να διαβάσει τους κανόνες του παιχνιδιού.
2) Τα διαθέσιμα χρώματα που υπάρχουν.
3) Ο κωδικός που πρέπει να βρει ο πράκτορας.
4) Οι δοκιμές του πράκτορα.
5) Τα αποτελέσματα των δοκιμών με τον κρυφό κωδικό. Το κόκκινο σημάδι σημαίνει ότι ένα χρώμα υπάρχει και βρίσκεται στην σωστή θέση και το λευκό, ότι υπάρχει αλλά δεν είναι στην σωστή θέση.
6) Με το κουμπί «Continue» εμφανίζεται η επόμενη δοκιμή του αλγορίθμου.
7) Το μέγεθος του κωδικού.
8) Το πλήθος των διαθέσιμων χρωμάτων.

9) Ο χρήστης μπορεί να επιλέξει αν θα υπάρχουν διπλότυπα χρώματα ή όχι.
10) Ο χρήστης μπορεί να διαλέξει ποιον αλγόριθμο θα ακολουθήσει ο πράκτορας. Σε περίπτωση που επιθυμεί να παίξει ο ίδιος, τότε πρέπει απλά να μην επιλέξει κανέναν αλγόριθμο.
11) Μέσω του κουμπιού «NewGame» ξεκινάει νέα παρτίδα παιχνιδιού.

Πρέπει να επισημανθεί, ότι για την διευκόλυνση των συγκρίσεων που χρειάζεται να πραγματοποιηθούν σε κάθε αλγόριθμο, έχει γίνει ανάθεση αριθμητικής τιμής σε κάθε χρώμα (Πίνακας 4.1). Με αυτό τον τρόπο ο πράκτορας αντιλαμβάνεται τον χρωματικό κωδικό, ως ένα κωδικό από αριθμητικές τιμές. Για παράδειγμα ο κωδικός «Κόκκινο, Κίτρινο, Κόκκινο, Μπλε» αντιλαμβάνεται από τον πράκτορα ως «1,2,1,4»

Πίνακας 4.1: Αναθέσεις Χρωμάτων

Χρώμα	**Αριθμητική Τιμή**
Κόκκινο	1
Κίτρινο	2
Πράσινο	3
Μπλε	4
Ροζ	5
Γαλάζιο	6
Πορτοκαλί	7
Μαύρο	8

4.2 Serial Algorithm

Ο πρώτος αλγόριθμος που κατασκευάστηκε είναι σχετικά απλός στην κατανόηση και στην υλοποίησή του, ενώ ταυτόχρονα δεν χρειάζεται η γνώση και η αξιοποίηση κάποιου άλλου αλγορίθμου. Προσπαθεί να εντοπίσει τον κρυφό κωδικό, δημιουργώντας σταδιακά όλους τους υποψήφιους κωδικούς και αποκλείει αυτούς που αδυνατούν να επιλύσουν το παιχνίδι. Με βάση την κάθε δοκιμή που επιχειρεί και το αποτέλεσμα που λαμβάνει, δημιουργεί πιθανά μοντέλα επίλυσης. Σε κάθε επανάληψη, ο αλγόριθμος διαλέγει να δοκιμάσει ένα χρώμα την φορά. Με αυτόν τον τρόπο μπορεί να το αποκλείσει από τις πιθανές λύσεις ή να καταλάβει ακριβώς πόσες φορές έχει χρησιμοποιηθεί, εφόσον βέβαια η χρήση του επιτρέπεται περισσότερες από μία φορά.

Καθώς το αρχικό παιχνίδι είχε δημιουργηθεί για διαθεσιμότητα έξι χρωμάτων και για κωδικούς των τεσσάρων, η ανάλυση και τα παραδείγματα που θα ακολουθήσουν θα αφορούν κι αυτά κωδικούς τεσσάρων χρωμάτων. Με την αύξηση στο μέγεθος του κωδικού ή των διαθέσιμων επιλογών, η λογική του αλγορίθμου δεν επηρεάζεται, αντ' αυτού όμως επηρεάζεται η πολυπλοκότητα και αυξάνεται η δυσκολία επίλυσης του παιχνιδιού.

Παρακάτω παρουσιάζεται αναλυτικότερα η λογική του αλγορίθμου, όπως και τα βήματα που ακολουθεί.

1. Πρώτα απ' όλα, ο αλγόριθμός αποθηκεύει στην μνήμη το μοντέλο «XXXX». Η τιμή του κάθε «X» μπορεί να είναι οποιοδήποτε από τα διαθέσιμα χρώματα. Έτσι αναγκαστικά ο κωδικός που ψάχνει να βρει θα είναι της παραπάνω μορφής. Δηλαδή
   - $διαθέσιμα\ χρώματα\ = \{1, 2, 3, 4, 5, 6\}$
   - $X \in διαθέσιμα\ χρώματα$
   - $αρχικό\ μοντέλο\ = \ll XXXX \gg$
   - $κωδικός\ \in πιθανά\ μοντέλα$
2. Την πρώτη φορά που θα τρέξει ο αλγόριθμος, θα διαλέξει τον κωδικό «1111», καθώς αντιπροσωπεύει και το πρώτο διαθέσιμο χρώμα που υπάρχει.
3. Ανάλογα με το αποτέλεσμα που θα λάβει, θα συμπληρώσει κατάλληλα τα πιθανά μοντέλα που υπάρχουν. Συνεπώς τα μοντέλα ή θα παραμείνουν ίδια ή θα επικαιροποιηθούν. Σε κάθε περίπτωση το χρώμα που δοκιμάστηκε θα αφαιρεθεί από τα διαθέσιμα χρώματα, καθώς οι πληροφορίες που σχετίζονται με αυτό αντικατοπτρίζονται πλέον στα μοντέλα.

4. Ταξινομεί τα μοντέλα που έχουν δημιουργηθεί σε αύξουσα σειρά. Δηλαδή το μοντέλο «1211» είναι μικρότερο από το μοντέλο «2111». Επιπλέον, το μοντέλο «1Χ11» είναι μεγαλύτερο από το μοντέλο «1211», καθώς για να έχει παραμείνει η τιμή «Χ» στο μοντέλο, σημαίνει ότι θα είναι αναγκαστικά μεγαλύτερη του «2».
5. Διαλέγει το μοντέλο με την μικρότερη τιμή και αντικαθιστά όλες τις τιμές «Χ» με το πρώτο διαθέσιμο και μη χρησιμοποιημένο χρώμα που υπάρχει.
6. Τα βήματα 3 έως 5 επαναλαμβάνονται, μέχρις ότου ο αλγόριθμος εντοπίσει όλα τα χρώματα που απαρτίζουν τον κωδικό.
7. Αφού εντοπίσει όλα τα πιθανά χρώματα, διαλέγει και πάλι τον μικρότερο αριθμητικά κωδικό και διαγράφει οποιονδήποτε κωδικό, που δεν θα έδινε το ίδιο αποτέλεσμα, εφόσον η τρέχουσα δοκιμή ήταν ο κρυφός κωδικός.

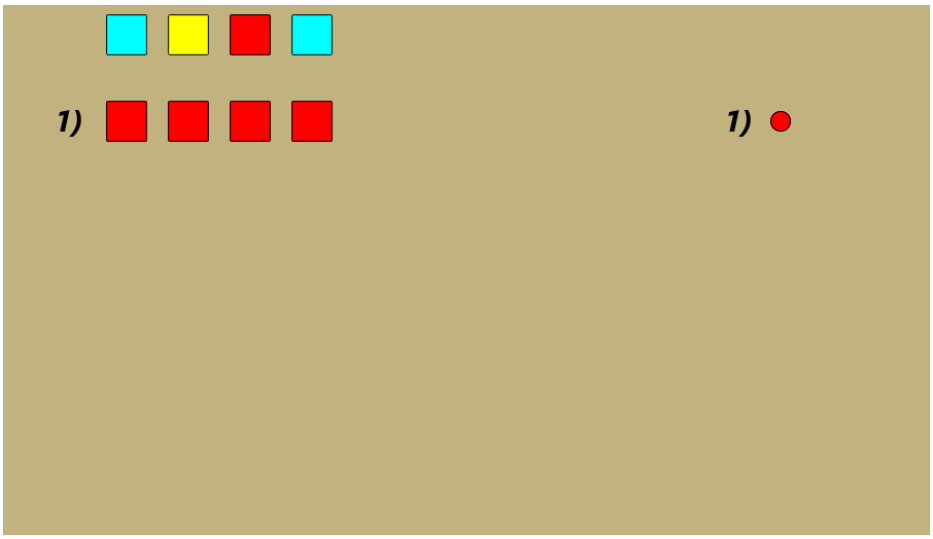
Παραδείγματος χάρη, εάν η δοκιμή ήταν ο κωδικός «1122», και το αποτέλεσμα που θα λαμβάναμε ήταν ΚΛ (ένα κόκκινο και ένα λευκό σημάδι), τότε:

- (1122, 1112) = *ΚΚΚ*, 3 νούμερα βρίσκονται στην ίδια θέση.
  ΚΚΚ ≠ *ΚΛ*: Ο συνδυασμός «1112» αφαιρείται.
- (1122, 1113) = ΚΚ, 2 νούμερα βρίσκονται στην ίδια θέση.
  ΚΚ ≠ *ΚΛ*: Ο συνδυασμός «1113» αφαιρείται.
- (1122, 1114) = *ΚΚ*, 2 νούμερα βρίσκονται στην ίδια θέση.
  ΚΚ ≠ *ΚΛ*: Ο συνδυασμός «1114» αφαιρείται.
- (1122, 1314) = *ΚΛ*, 1 νούμερο βρίσκεται στην ίδια θέση και ένα σε διαφορετική.
  ΚΛ = *ΚΛ*: Ο συνδυασμός «1314» παραμένει.

Η διαδικασία αυτή επαναλαμβάνεται έως ότου βρεθεί η λύση του παιχνιδιού.

Στην περίπτωση, που δεν επιτρέπεται η χρήση των χρωμάτων πολλές φορές, ο αρχικός κωδικός που επιλέγεται είναι ο «1234». Και ακολουθείται η ίδια διαδικασία, όπως προηγουμένως. Δηλαδή, ανάλογα με το αποτέλεσμα, συμπληρώνονται κατάλληλα τα πιθανά μοντέλα και με την παραδοχή και πάλι ότι προτιμώνται τα χρώματα και οι κωδικοί με τη μικρότερη αριθμητική τιμή, γίνεται αντίστοιχα η κάθε επιλογή.

Για την καλύτερη κατανόηση του αλγορίθμου, ακολουθεί ένα παράδειγμα, όπου η επιλογή διπλών χρωμάτων επιτρέπεται και ο κρυφός κωδικός είναι ο «6216».



Εικόνα 4.3: Serial Algorithm – Βήμα 1

Ο πρώτος συνδυασμός που χρησιμοποιείται είναι ο «1111». Από το αποτέλεσμα που λαμβάνει, ένα κόκκινο σημάδι, τα διαθέσιμα πιθανά μοντέλα που προκύπτουν είναι τα εξής:

«1XXX», «X1XX», «XX1X», «XXX1»

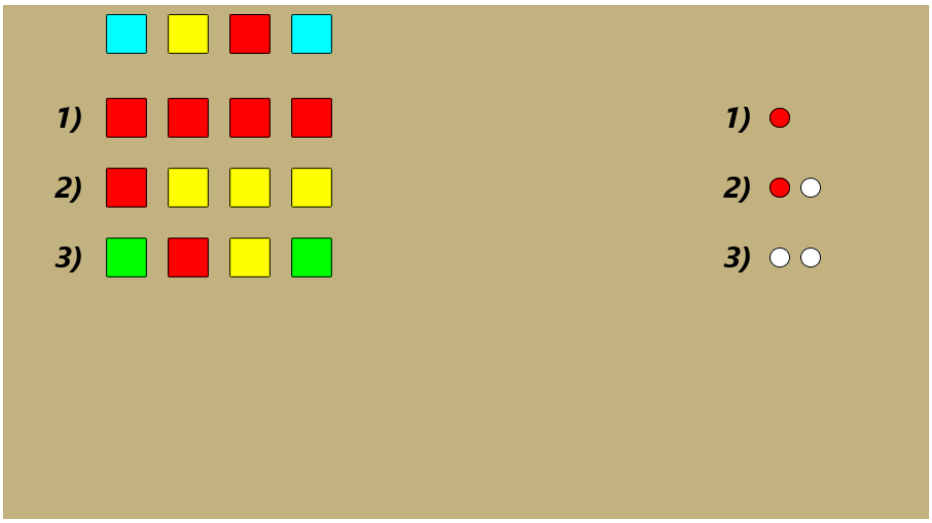
Με το κόκκινο χρώμα δεν χρειάζεται να ασχοληθεί περαιτέρω, καθώς αναγκαστικά θα βρίσκεται σε κάποια από τις προ-αναφερθείσες θέσεις.

Εικόνα 4.4: Serial Algorithm – Βήμα 2

Διαλέγοντας το μικρότερο μοντέλο, δηλαδή το «1XXX», τα άγνωστα χρώματα συμπληρώνονται από το αμέσως επόμενο χρώμα, το κίτρινο, που του έχει τεθεί η τιμή «2». Από το αποτέλεσμα της δοκιμής μπορούμε να συμπεράνουμε τα εξής. Το «Κόκκινο» δεν

μπορεί να είναι στην πρώτη θέση, καθώς για να επαληθεύεται το αποτέλεσμα θα έπρεπε το «Κίτρινο» να τοποθετηθεί σε κάποια διαφορετική θέση. Η μόνη διαθέσιμη είναι η πρώτη, όπου έχει τοποθετηθεί το «Κόκκινο». Έτσι καταλήγουμε σε άτοπο και ο πράκτορας συνειδητοποιεί ότι κάποιο «Κίτρινο» είναι στην σωστή θέση, ενώ το «Κόκκινο» όχι. Όλα τα παραπάνω αποτυπώνονται στα πιθανά μοντέλα που έχουν δημιουργηθεί, τα οποία είναι τα εξής:

«X12X», «X1X2», «X21X», «X2X1», «XX12», «XX21»



Εικόνα 4.5: Serial Algorithm – Βήμα 3

Το μοντέλο που επιλέγεται να συμπληρωθεί τώρα είναι το «X12X», που είναι και το μικρότερο. Το επόμενο χρώμα που θα δοκιμαστεί είναι το «Πράσινο», που του έχει ανατεθεί και η τιμή «3». Από την στιγμή που δεν αλλάζει ο συνολικός αριθμός των σημαδιών του αποτελέσματος, συνεπάγεται ότι το πράσινο χρώμα δεν υπάρχει στον κρυφό κωδικό. Επιπλέον έχοντας ως αποτέλεσμα δύο λευκά σημάδια, μπορούμε να αφαιρέσουμε από τα μοντέλα, οποιοδήποτε έχει στην δεύτερη θέση το «Κόκκινο» και στην τρίτη το «Κίτρινο». Συνεπώς τα διαθέσιμα μοντέλα έχουν διαμορφωθεί πλέον ως εξής:

«X21X», «X2X1», «XX12»

Εικόνα 4.6: Serial Algorithm – Βήμα 4

Διαλέγοντας το μοντέλο «X21X» και τοποθετώντας το επόμενο διαθέσιμο χρώμα, το «Μπλε», λαμβάνουμε ως αποτέλεσμα δύο κόκκινα σημάδια. Ο συνολικός αριθμός των σημαδιών του αποτελέσματος παραμένει ίδιος και ως εκ τούτου το «Μπλε» αποκλείεται. Έτσι, το «Κίτρινο» χρώμα υποχρεωτικά βρίσκεται στην δεύτερη θέση, όπως και το «Κόκκινο» στην τρίτη. Πλέον το μοναδικό πιθανό μοντέλο είναι το «X21X».

Εικόνα 4.7: Serial Algorithm – Βήμα 5

Διαλέγοντας το μοναδικό μοντέλο «X21X» και τοποθετώντας το «Ροζ», εξάγουμε το ίδιο αποτέλεσμα με προηγουμένως. Συνεπώς, στην ύπαρξη των μοντέλων δεν έχει αλλάξει πλέον κάτι, καθώς ο αλγόριθμος ήδη από το προηγούμενο βήμα έχει εντοπίσει τις

θέσεις του «Κόκκινου» και του «Κίτρινου» χρώματος. Το μοναδικό μοντέλο παραμένει το «X21X».

Εικόνα 4.8: Serial Algorithm – Βήμα 6

Τέλος, έχοντας δοκιμάσει όλα τα πιθανά χρώματα εκτός του «Γαλάζιου», που είναι και το τελευταίο, εξάγουμε τέσσερα κόκκινα σημάδια, που μας υποδεικνύει ότι έχουμε εντοπίσει τον κρυφό κωδικό, που δεν είναι άλλος από τον «6216».

Αξίζει να τονιστεί ότι ο συγκεκριμένος αλγόριθμος και η εσωτερική διαχείριση μοντέλων, τα οποία κωδικοποιούν πολλαπλούς συνδυασμούς, οδηγεί σε σημαντική εξοικονόμηση μνήμης.

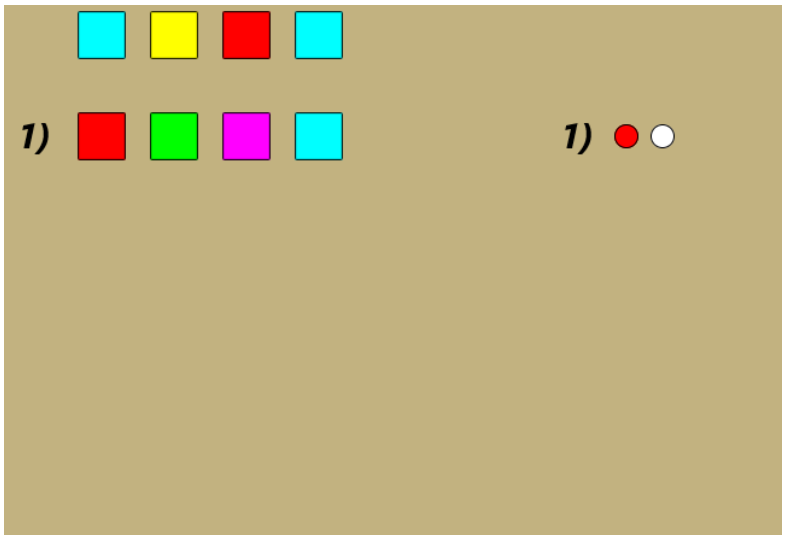
4.3 Simple Algorithm

Ο δεύτερος αλγόριθμος που υλοποιήθηκε είναι και αυτός σχετικά απλός στην υλοποίησή του. Όπως και ο προηγούμενος, δεν απαιτεί κάποια περαιτέρω γνώση πάνω στους ευρέως γνωστούς αλγορίθμους της τεχνητής νοημοσύνης. Σε αντίθεση με τον προηγούμενο, ο «Simple Algorithm» δημιουργεί αρχικά ένα σύνολο από όλους τους πιθανούς συνδυασμούς και σε κάθε γύρο παιχνιδιού, αφαιρεί όλες τις επιλογές, οι οποίες είναι αδύνατο να επιλύουν το παιχνίδι. Σε κάθε γύρο, η επιλογή της δοκιμής γίνεται τυχαία και αυτό κάνει τον αλγόριθμο μη ντετερμινιστικό. Έτσι δεν θα προκύπτουν πάντα τα ίδια αποτελέσματα για τον εκάστοτε κωδικό.

Πιο αναλυτικά, τα βήματα που ακολουθεί ο «Simple Algorithm» είναι τα παρακάτω:

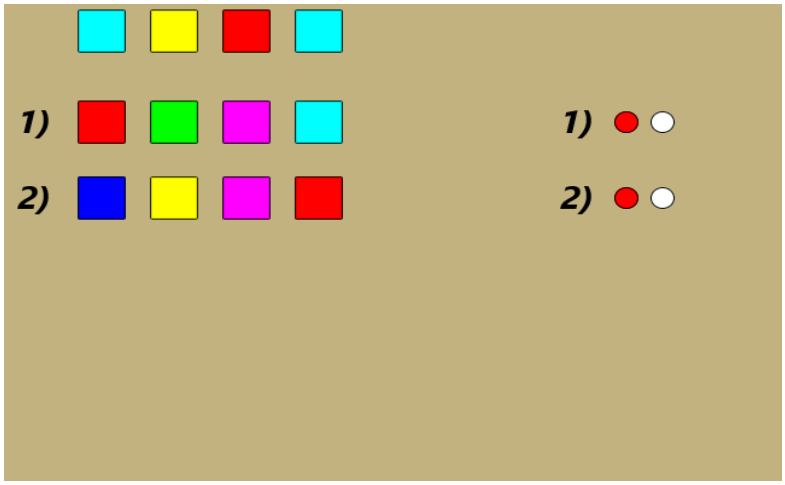
1. Σαν πρώτο βήμα καλείται να υπολογίσει και να αποθηκεύσει στη μνήμη όλους τους πιθανούς συνδυασμούς που μπορούν να προκύψουν.
2. Ταξινομεί όλους τους διαθέσιμους συνδυασμούς σε αύξουσα σειρά.
3. Διαλέγει τυχαία έναν από τους διαθέσιμους κωδικούς.
4. Λαμβάνει το αποτέλεσμα της δοκιμής και του κρυφού κωδικού.
5. Εάν η πληροφορία που λαμβάνει είναι τέσσερα κόκκινα σημάδια το παιχνίδι έχει ολοκληρωθεί και ο αλγόριθμος τερματίζεται.
6. Διαγράφει οποιονδήποτε κωδικό, που δεν θα έδινε το ίδιο αποτέλεσμα, εφόσον η τρέχουσα δοκιμή ήταν ο κρυφός κωδικός.
7. Τα βήματα 2-6 επαναλαμβάνονται μέχρι να βρεθεί ο κρυφός κωδικός και να ολοκληρωθεί το παιχνίδι.

Παρακάτω παρουσιάζεται ένα παράδειγμα χρήσης του αλγορίθμου, για την ορθότερη κατανόηση. Ο κρυφός κωδικός είναι ο «6216» και η πολλαπλή χρήση των χρωμάτων επιτρέπεται.



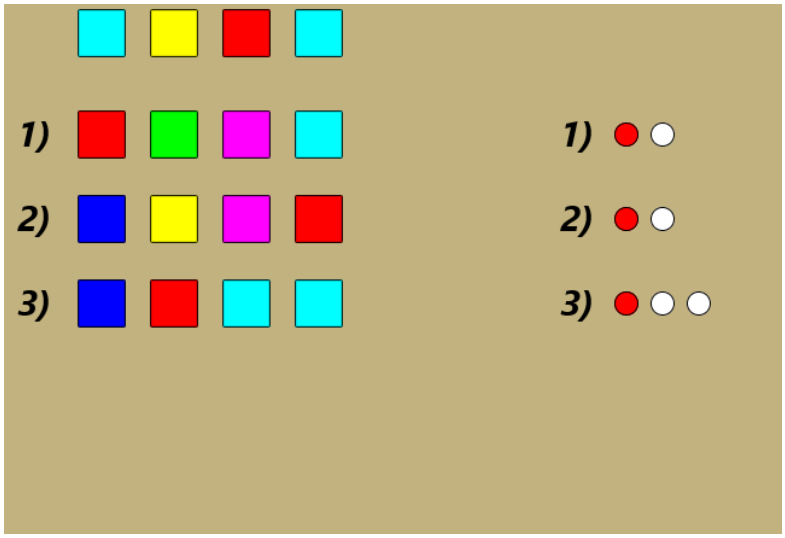
Εικόνα 4.9: Simple Algorithm – Βήμα 1

Η πρώτη δοκιμή που πραγματοποιείται είναι ο κωδικός «1356», ο οποίος έχει επιλεχθεί τυχαία. Ύστερα ο πράκτορας θα αποκλείσει όλους τους κωδικούς, οι οποίοι αν συγκριθούν με την δοκιμή, δηλαδή με τον συνδυασμό «1356», δεν θα δώσουν το ίδιο αποτέλεσμα. Θα παραμείνουν εκείνοι μόνο που θα έδιναν την ίδια έξοδο. Με αυτόν τον τρόπο, από τους 1296 κωδικούς που υπήρχαν αρχικά, πλέον θα είναι διαθέσιμοι οι 252 από αυτούς.



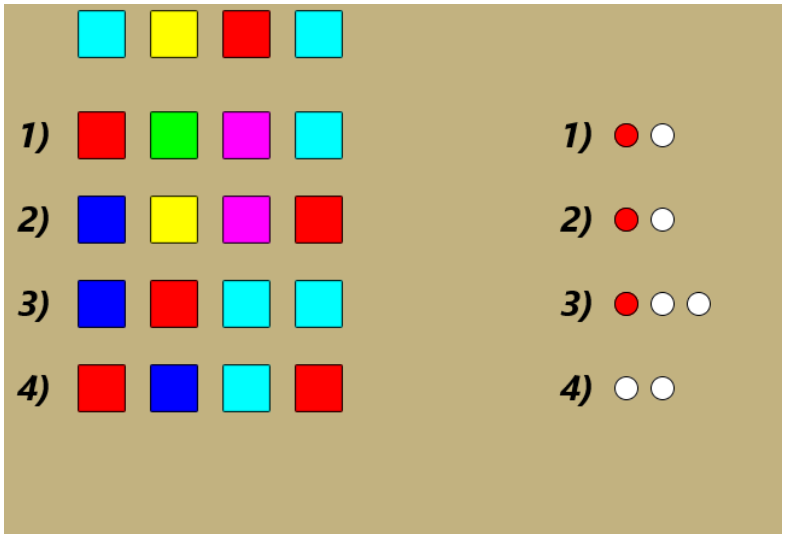
Εικόνα 4.10: Simple Algorithm – Βήμα 2

Με τον ίδιο τρόπο όπως προηγουμένως, επιλέγεται τυχαία ο κωδικός «4251». Ο πράκτορας θα αφαιρέσει από την μνήμη, όλους τους κωδικούς εκείνους, που δεν θα έδιναν ως αποτέλεσμα ένα κόκκινο και ένα λευκό σημάδι, εάν η δοκιμή ήταν ο κρυφός κωδικός. Από τους 252 κωδικούς που είχαν απομείνει από πριν, τώρα είναι διαθέσιμοι μόλις 50.



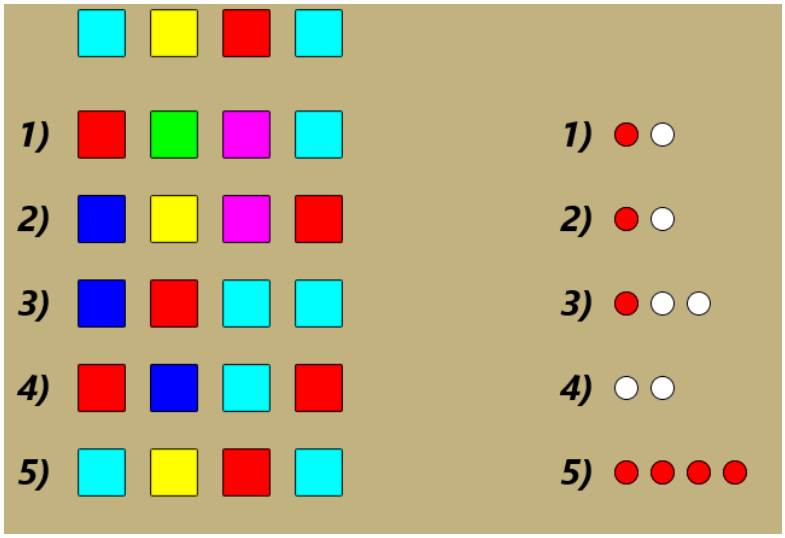
Εικόνα 4.11: Simple Algorithm – Βήμα 3

Αντίστοιχα, η επόμενη δοκιμή που θα πραγματοποιηθεί είναι ο κωδικός «4166». Όσοι κωδικοί συγκριθούν με αυτόν και δεν δώσουν αποτέλεσμα ένα κόκκινο και δύο λευκά σημάδια θα διαγραφούν. Οι κωδικοί που έχουν απομείνει είναι πλέον 2.



Εικόνα 4.12: Simple Algorithm – Βήμα 4

Η επόμενη δοκιμή είναι κωδικός «1461». Θα αφαιρεθούν όσοι κωδικοί συγκριθούν με τον «1461» και δεν δώσουν αποτέλεσμα δύο λευκά σημάδια Αυτό έχει ως αποτέλεσμα, από τους 1296 κωδικούς που υπήρχαν αρχικά, πλέον να έχει απομείνει μόνο ένας από αυτούς, ο «6216».



Εικόνα 4.13: Simple Algorithm – Βήμα 5

Τέλος, θα δοκιμαστεί ο μοναδικός συνδυασμός που έχει απομείνει, ο «6216». Η λύση επιβεβαιώνεται, αφού ως αποτέλεσμα έχουμε 4 κόκκινα σημάδια. Ο ζητούμενος κωδικός έχει βρεθεί με την χρήση πέντε προσπαθειών.

4.4 Knuth Algorithm

Το 1977 ο Αμερικανός επιστήμονας Donald Knuth, ο οποίος θεωρείται ο «πατέρας της ανάλυσης των αλγορίθμων», κατασκεύασε έναν αλγόριθμο, ο οποίος μπορεί να εντοπίσει τον κρυφό κωδικό με την χρήση το πολύ πέντε προσπαθειών, για την κλασσική έκδοση του παιχνιδιού (κωδικός των τεσσάρων χρωμάτων και διαθεσιμότητα έξι χρωμάτων). Η λογική της συγκεκριμένης στρατηγικής, είναι να πραγματοποιήσει ελαχιστοποίηση στην χείριστη περίπτωση. Προτού, γίνει όμως η ανάλυση του αλγορίθμου, είναι απαραίτητο να δοθεί η επεξήγηση του αλγορίθμου minimax, καθώς αποτελεί βασικό κομμάτι [36].

Ο αλγόριθμος minimax είναι ένας κανόνας απόφασης που χρησιμοποιείται στην τεχνητή νοημοσύνη, τη θεωρία αποφάσεων, τη θεωρία παιγνίων, τη στατιστική και τη φιλοσοφία για την ελαχιστοποίηση της πιθανής απώλειας σε ένα σενάριο χειρότερης περίπτωσης (μέγιστης απώλειας). Αρχικά, διατυπώθηκε για τη θεωρία παιγνίων μηδενικού αθροίσματος πολλών παικτών, καλύπτοντας τόσο τις περιπτώσεις όπου οι παίκτες κάνουν εναλλαγή στις κινήσεις, όσο και εκείνες όπου κάνουν ταυτόχρονες κινήσεις, επεκτάθηκε όμως και σε πιο σύνθετα παιχνίδια και στη γενική λήψη αποφάσεων με αβεβαιότητα. Ο minimax είναι πολύ δημοφιλής στον τομέα της τεχνητής νοημοσύνης και ο λόγος είναι ότι λαμβάνει υπόψη όλες τις πιθανές κινήσεις που μπορούν να κάνουν οι παίκτες οποιαδήποτε στιγμή κατά τη διάρκεια του παιχνιδιού. Με αυτές τις πληροφορίες, επιχειρεί στη συνέχεια να ελαχιστοποιήσει το πλεονέκτημα του αντίπαλου παίκτη, ενώ μεγιστοποιεί το πλεονέκτημα του πράκτορα.

Ο τρόπος που χρησιμοποιείται στην περίπτωση της παρούσας εργασίας, όπως και ολόκληρος αλγόριθμος αναλύεται παρακάτω.

1. Αρχικά δημιουργείται ένα σετ S από όλους τους πιθανούς κωδικούς.
2. Σαν πρώτη δοκιμή, ξεκινάει με τον κωδικό «1122».
3. Λαμβάνει το αποτέλεσμα της δοκιμής και του κρυφού κωδικού.
4. Εάν η πληροφορία που λαμβάνει είναι τέσσερα κόκκινα σημάδια το παιχνίδι έχει ολοκληρωθεί και ο αλγόριθμος τερματίζεται.
5. Διαφορετικά, διαγράφει οποιονδήποτε κωδικό, που δεν θα έδινε το ίδιο αποτέλεσμα, εφόσον η τρέχουσα δοκιμή ήταν ο κρυφός κωδικός.
6. Εφαρμόζει τον αλγόριθμο minimax, προκειμένου να επιλέξει την επόμενη δοκιμή. Για οποιοδήποτε αχρησιμοποίητο κωδικό και όχι μόνο για εκείνους που ανήκουν

στο S, υπολογίζει πόσους κωδικούς μπορεί να εξαλείψει από το S, για κάθε πιθανό αποτέλεσμα που μπορεί να παραχθεί. Έτσι, δημιουργείται ένα σύνολο δοκιμών με την μικρότερη μέγιστη βαθμολογία. Από το νέο σύνολο που έχει προκύψει, επιλέγεται η νέα δοκιμή, προτιμώντας αυτή να είναι μέλος τους S, όποτε είναι δυνατόν.

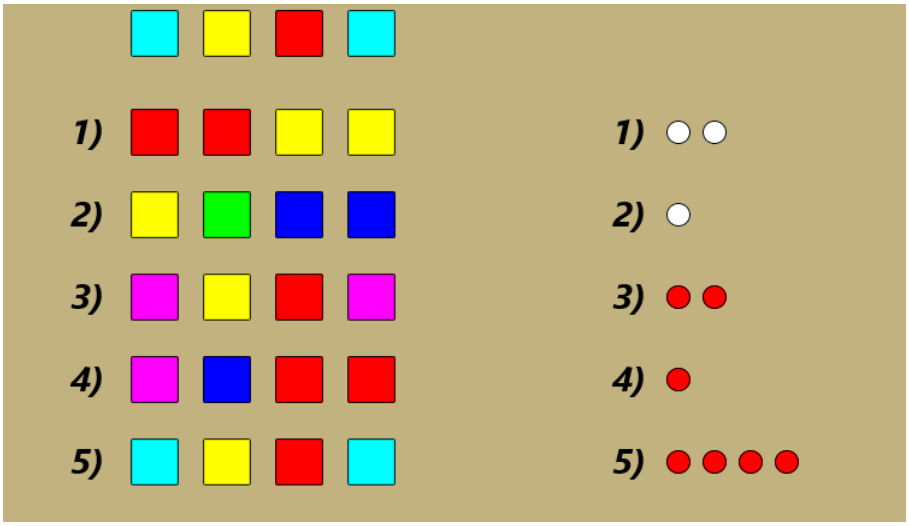
Ο αλγόριθμος ακολουθεί την σύμβαση της επιλογής του κωδικού με την μικρότερη αριθμητική τιμή. Π.χ. το «2345» είναι μικρότερο από το «3456».

7. Ο αλγόριθμος επαναλαμβάνεται από το βήμα 3.

Αξίζει να σημειωθεί πως ο λόγος που επιλέχτηκε ως πρώτη κίνηση ο κωδικός «1122» δεν είναι τυχαίος. Όπως και στην πορεία του αλγορίθμου, έτσι και στην αρχή του παιχνιδιού, σκοπός είναι να επιλέγεται ένας κωδικός που ελαχιστοποιεί την χείριστη κατάσταση. Έτσι, με βάση τον Πίνακα 4.3, έγινε και η επιλογή της πρώτης προσπάθειας.

Επιπλέον πρέπει να σημειωθεί ότι ο λόγος που λαμβάνονται όλοι οι κωδικοί στον minimax αλγόριθμο, είναι επειδή με αυτόν τον τρόπο προσπαθούμε να μικρύνουμε το πλήθος των διαθέσιμων συνδυασμών και όχι να εντοπίσουμε κατευθείαν ποιος είναι ο σωστός.

Για την καλύτερη κατανόηση του αλγορίθμου, παρουσιάζεται ένα παράδειγμα, όπου ο κρυφός κωδικός είναι ο «6216» και η πολλαπλή χρήση των χρωμάτων επιτρέπεται.



Εικόνα 4.14: Knuth Algorithm – Βήματα Επίλυσης

Η πρώτη δοκιμή που πραγματοποιείται είναι πάντα ίδια και είναι η «1122». Ύστερα ο αλγόριθμος θα αποκλείσει οποιοδήποτε κωδικό που αν συγκριθεί με την δοκιμή, θα δώσει διαφορετικό αποτέλεσμα. Προκειμένου να πραγματοποιηθεί η επόμενη δοκιμή, ο πράκτορας θα προσπαθήσει για κάθε αχρησιμοποίητο κωδικό να εντοπίσει πόσους κωδικούς από τους εναπομείναντες μπορεί να εξαλείψει για κάθε πιθανό αποτέλεσμα. Για κάθε αχρησιμοποίητο κωδικό κρατάει την χειρότερη περίπτωση και στο τέλος διαλέγει τον κωδικό εκείνο με την μικρότερη χείριστη κατάσταση. Αν υπάρχουν περισσότεροι του ενός κωδικού, τότε προτιμάται ο κωδικός με την μικρότερη αριθμητική τιμή.

Όπως αναφέρθηκε και προηγουμένως, η επιλογή της πρώτης δοκιμής δεν πραγματοποιείται τυχαία. Ο Πίνακας 4.2 παρουσιάζει για κάθε επιλογή, πόσοι κωδικοί θα παρέμεναν, σύμφωνα με το ανάλογο αποτέλεσμα ή διαφορετικά πόσοι κωδικοί ικανοποιούν την συνθήκη του αποτελέσματος. Οι κωδικοί που παρουσιάζονται είναι όλοι οι πιθανοί συνδυασμοί που μπορούν να προκύψουν για κωδικούς των 4 χρωμάτων, αν αναλογιστεί κανείς, πως και εδώ εφαρμόζεται η σύμβαση της επιλογής του κωδικού με την μικρότερη αριθμητική τιμή.

Πίνακας 4.2: Knuth Algorithm - Αποτελέσματα 1ης Δοκιμής

RESULTS	1111	1112	1122	1123	1234
(0,0)	**625**	256	**256**	81	16
(0,1)	0	308	**256**	**276**	152
(0,2)	0	61	96	222	**312**
(0,3)	0	0	16	44	136
(0,4)	0	0	1	2	9
(1,0)	500	**317**	**256**	182	108
(1,1)	0	156	208	230	252
(1,2)	0	27	36	84	132
(1,3)	0	0	0	4	8
(2,0)	150	123	114	105	96
(2,1)	0	24	32	40	48
(2,2)	0	3	4	5	6
(3,0)	20	20	20	20	20
(4,0)	1	1	1	1	1

Πίνακας 4.3: Knuth Algorithm - Συνολικά Αποτελέσματα

ΚΩΔΙΚΟΣ	ΧΕΙΡΟΤΕΡΗ ΚΑΤΑΣΤΑΣΗ
1111	625
1112	317
1122	**256**
1123	276
1234	312

Ο Πίνακας 4.3 φανερώνει την χειρότερη κατάσταση για την κάθε περίπτωση. Αφού επιθυμούμε να ελαχιστοποιήσουμε την χείριστη κατάσταση, επιλέγεται ο κωδικός «1122».

Αντίστοιχα έχουν υπολογιστεί για κάθε επίπεδο του παιχνιδιού οι πρώτες δοκιμές που θα πραγματοποιηθούν, οι οποίες παρουσιάζονται παρακάτω:

Διαθέσιμα Χρώματα 8, Μήκος Κωδικού 4

Πίνακας 4.4: Knuth Algorithm - Αποτελέσματα 1ης Δοκιμής

RESULTS	1111	1112	1122	1123	1234
(0,0)	**2401**	**1296**	**1296**	625	256
(0,1)	0	978	864	**1160**	**976**
(0,2)	0	127	216	546	936
(0,3)	0	0	24	68	224
(0,4)	0	0	1	2	9
(1,0)	1372	991	864	682	500
(1,1)	0	342	456	558	660
(1,2)	0	39	52	128	204
(1,3)	0	0	0	4	8
(2,0)	294	255	242	229	216
(2,1)	0	36	48	60	72
(2,2)	0	3	4	5	6
(3,0)	28	28	28	28	28
(4,0)	1	1	1	1	1

Πίνακας 4.5: Knuth Algorithm - Συνολικά Αποτελέσματα

ΚΩΔΙΚΟΣ	ΧΕΙΡΟΤΕΡΗ ΚΑΤΑΣΤΑΣΗ
1111	2401
1112	1296
1122	1296
1123	1160
1234	**976**

Σαν πρώτη δοκιμή επιλέγεται ο κωδικός «1234», που εξασφαλίζει και το μικρότερο πλήθος κωδικών.

Διαθέσιμα Χρώματα 6, Μήκος Κωδικού 5

Πίνακας 4.6: Knuth Algorithm - Αποτελέσματα 1ης Δοκιμής

RESULTS	11111	11112	11122	11123	11223	11234	12345
(0,0)	**3125**	1024	1024	243	243	32	1
(0,1)	0	**1732**	1280	1212	1011	438	75
(0,2)	0	369	656	1306	1190	**1303**	770
(0,3)	0	0	152	340	574	1079	1550
(0,4)	0	0	13	24	103	261	685
(0,5)	0	0	0	0	4	12	44
(1,0)	**3125**	1649	1280	755	580	275	80
(1,1)	0	1232	**1384**	**1444**	**1360**	1132	760
(1,2)	0	244	410	788	984	1290	**1560**
(1,3)	0	0	48	132	192	404	680
(1,4)	0	0	3	6	9	24	45
(2,0)	1250	884	701	590	492	381	270
(2,1)	0	312	468	501	564	597	630
(2,2)	0	54	81	153	186	258	330
(2,3)	0	0	0	6	8	14	20
(3,0)	250	214	196	187	178	169	160
(3,1)	0	32	48	56	64	72	80
(3,2)	0	4	6	7	8	9	10
(4,0)	25	25	25	25	25	25	25
(5,0)	1	1	1	1	1	1	1

Πίνακας 4.7: Knuth Algorithm - Συνολικά Αποτελέσματα

ΚΩΔΙΚΟΣ	ΧΕΙΡΟΤΕΡΗ ΚΑΤΑΣΤΑΣΗ
11111	3125
11112	1732
11122	1384
11123	1444
11223	1360
11234	**1303**
12345	1560

Σαν πρώτη δοκιμή επιλέγεται ο κωδικός «11234», που εξασφαλίζει και το μικρότερο πλήθος κωδικών.

Διαθέσιμα Χρώματα 7, Μήκος Κωδικού 5

Πίνακας 4.8: Knuth Algorithm - Αποτελέσματα 1ης Δοκιμής

RESULTS	11111	11112	11122	11123	11223	11234	12345
(0,0)	**7776**	3125	**3125**	1024	1024	243	32
(0,1)	0	**3980**	**3125**	**3464**	**3012**	1818	650
(0,2)	0	671	1275	2714	2642	**3402**	2840
(0,3)	0	0	235	544	962	1956	**3260**
(0,4)	0	0	16	30	132	345	950
(0,5)	0	0	0	0	4	12	44
(1,0)	6480	3796	**3125**	2018	1649	930	405
(1,1)	0	2320	2660	3008	2964	2736	2220
(1,2)	0	364	632	1280	1614	2262	2910
(1,3)	0	0	60	168	244	528	900
(1,4)	0	0	3	6	9	24	45
(2,0)	2160	1614	1341	1158	1006	823	640
(2,1)	0	480	720	804	912	996	1080
(2,2)	0	66	99	192	234	327	420
(2,3)	0	0	0	6	8	14	20
(3,0)	360	316	294	283	272	261	250
(3,1)	0	40	60	70	80	90	100
(3,2)	0	4	6	7	8	9	10
(4,0)	30	30	30	30	30	30	30
(5,0)	1	1	1	1	1	1	1

Πίνακας 4.9: Knuth Algorithm - Αποτελέσματα 1ης Δοκιμής

ΚΩΔΙΚΟΣ	ΧΕΙΡΟΤΕΡΗ ΚΑΤΑΣΤΑΣΗ
11111	7776
11112	3980
11122	3125
11123	3464
11223	**3012**
11234	3402
12345	3260

Σαν πρώτη δοκιμή επιλέγεται ο κωδικός «11223», που εξασφαλίζει και το μικρότερο πλήθος κωδικών.

4.5 Expected Size Algorithm

Η απόφαση επιλογής κωδικού, μπορεί να βασίζεται στην αναμενόμενη κατάσταση αντί για την χείριστη, που περιγράφεται στον αλγόριθμο του Knuth. Συνεπώς, σύμφωνα με τον Robert Irving, θα μπορούσαμε να επικεντρωθούμε στο αναμενόμενο πλήθος υποψηφίων κωδικών που θα παρέμενε για κάθε δυνατή επιλογή [37]. Το αναμενόμενο μέγεθος ενός τέτοιου διαχωρισμού, είναι η πιθανότητα να παραχθεί το συγκεκριμένο αποτέλεσμα, πολλαπλασιασμένο με το μέγεθος του διαχωρισμού. Δηλαδή, αν ορίσουμε ως A το σετ όλων των πιθανών αποτελεσμάτων α_i και α(x,g) τη συνάρτηση που μας δίνει το αποτέλεσμα του συνδυασμού x με τον κωδικό g, τότε το αναμενόμενο μέγεθος για την πρώτη προσπάθεια του παιχνιδιού, για κωδικό p χρωμάτων από ένα σύνολο C χρωμάτων, ορίζεται ως εξής:

$$\sum_{\alpha_i \in A} P_g(\alpha_i) \cdot \#(\{x | x \in C^p \wedge \alpha(x, g) = \alpha_i\})$$

Η πιθανότητα η απάντηση στον κωδικό g να είναι α_i είναι η εξής

$$P_g(\alpha_i) = \frac{\#(\{x | x \in C^p \wedge \alpha(x, g) = \alpha_i\})}{\#(C^p)}$$

Έτσι προκύπτει ότι:

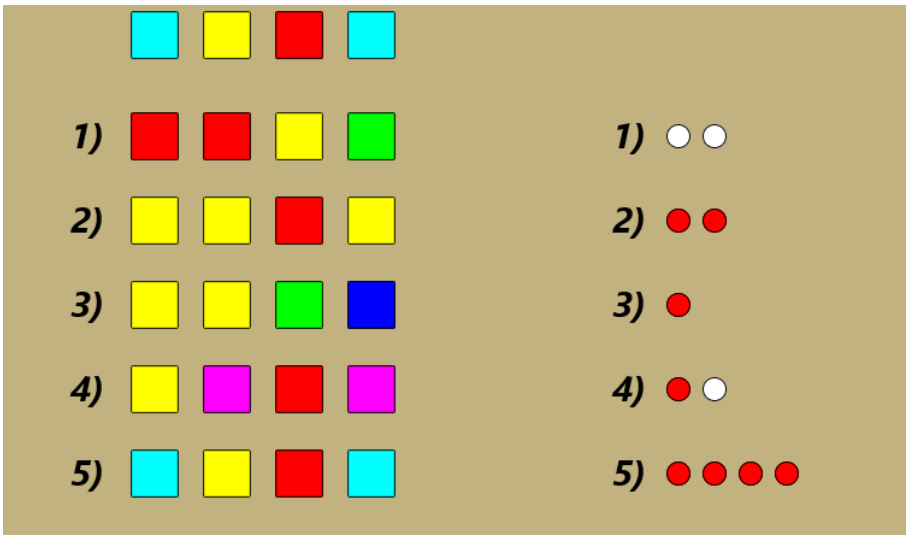
$$E(g) = \sum_{\alpha_i \in A} \frac{\#(\{x | x \in C^p \wedge \alpha(x, g) = \alpha_i\})^2}{\#(C^p)}$$

Με βάση τα παραπάνω, τα βήματα που πρέπει να ακολουθήσει ο αλγόριθμος είναι τα παρακάτω:

1. Αρχικά δημιουργείται ένα σετ S από όλους τους πιθανούς κωδικούς.
2. Σαν πρώτη δοκιμή, ξεκινάει με τον κωδικό «1123».
3. Λαμβάνει το αποτέλεσμα της δοκιμής και του κρυφού κωδικού.

4. Εάν η πληροφορία που λαμβάνει είναι τέσσερα κόκκινα σημάδια το παιχνίδι έχει ολοκληρωθεί και ο αλγόριθμος τερματίζεται.
5. Διαφορετικά, διαγράφει οποιονδήποτε κωδικό, που δεν θα έδινε το ίδιο αποτέλεσμα, εφόσον η τρέχουσα δοκιμή ήταν ο κρυφός κωδικός.
6. Υπολογίζει το αναμενόμενο μέγεθος με βάση την παραπάνω εξίσωση και επιλέγει αυτή με το μικρότερο μέγεθος.
7. Ο αλγόριθμος επαναλαμβάνεται από το βήμα 3.

Για την καλύτερη κατανόηση του παραπάνω αλγορίθμου, παρουσιάζεται το παράδειγμα που έχει αξιοποιηθεί και στους προηγούμενους αλγορίθμους. Ο ζητούμενος κωδικός είναι ο «6216».



Εικόνα 4.15: Expected Size Algorithm – Βήματα Επίλυσης

Η πρώτη δοκιμή που πραγματοποιείται είναι ο κωδικός «1123», ο οποίος αποτελεί πάντα την πρώτη δοκιμή. Στην συνέχεια, αφού παραμείνουν μόνο οι κωδικοί που μπορούν να επαληθεύσουν το παιχνίδι, ο πράκτορας υπολογίζει το αναμενόμενο μέγεθος για κάθε πιθανό συνδυασμό και επιλέγει αυτόν που του παρέχει το μικρότερο μέγεθος. Η διαδικασία αυτή επαναλαμβάνεται έως ότου ολοκληρωθεί το παιχνίδι.

Όπως και στον προηγούμενο αλγόριθμο, έτσι και τώρα η επιλογή της πρώτης δοκιμής δεν πραγματοποιείται με τυχαίο τρόπο, αλλά ακολουθεί την λογική της ευρετικής συνάρτησης. Έτσι έχει υπολογιστεί για κάθε επίπεδο το αναμενόμενο πλήθος κωδικών για κάθε πιθανή πρώτη δοκιμή.

Διαθέσιμα Χρώματα 6, Μήκος Κωδικού 4

Πίνακας 4.10: Expected Size Algorithm - Αποτελέσματα 1ης Δοκιμής

ΚΩΔΙΚΟΣ	ΑΝΑΜΕΝΟΜΕΝΟ ΜΕΓΕΘΟΣ
1111	511.9
1112	235.9
1122	204.5
1123	**185.3**
1234	188.2

Επιλέγεται ο κωδικός «1123», καθώς παρέχει το μικρότερο αναμενόμενο μέγεθος.

Διαθέσιμα Χρώματα 8, Μήκος Κωδικού 4

Πίνακας 4.11: Expected Size Algorithm - Αποτελέσματα 1ης Δοκιμής

ΚΩΔΙΚΟΣ	ΑΝΑΜΕΝΟΜΕΝΟ ΜΕΓΕΘΟΣ
1111	1888.3
1112	932.6
1122	852.6
1123	705.3
1234	**665.1**

Επιλέγεται ο κωδικός «1234», καθώς παρέχει το μικρότερο αναμενόμενο μέγεθος.

Διαθέσιμα Χρώματα 6, Μήκος Κωδικού 5

Πίνακας 4.12: Expected Size Algorithm - Αποτελέσματα 1ης Δοκιμής

ΚΩΔΙΚΟΣ	ΑΝΑΜΕΝΟΜΕΝΟ ΜΕΓΕΘΟΣ
11111	2712.75
11112	1204.28
11122	975.41
11123	934.89
11223	**852.39**
11234	885.08
12345	969.71

Επιλέγεται ο κωδικός «11223», καθώς παρέχει το μικρότερο αναμενόμενο μέγεθος.

Διαθέσιμα Χρώματα 7, Μήκος Κωδικού 5

Πίνακας 4.13: Expected Size Algorithm - Αποτελέσματα 1ης Δοκιμής

ΚΩΔΙΚΟΣ	ΑΝΑΜΕΝΟΜΕΝΟ ΜΕΓΕΘΟΣ
11111	6373.72
11112	2304.92
11122	2426.82
11123	2232.86
11223	**2030.04**
11234	2047.66
12345	2151.34

Επιλέγεται ο κωδικός «11223», καθώς παρέχει το μικρότερο αναμενόμενο μέγεθος.

4.6 Maximum Entropy Algorithm

Ο αλγόριθμος της μέγιστης εντροπίας αναπτύχθηκε αρχικά από τους Azer Bestavros και Ahmed Belal και παρουσιάστηκε τον Σεπτέμβριο του 1986 [38]. Σύμφωνα με την ακόλουθη στρατηγική, ο παίκτης (αποκωδικοποιητής) επιλέγει τον κωδικό, ο οποίος κατά μέσο όρο, προσφέρει την μέγιστη πιθανή ποσότητα πληροφορίας. Δηλαδή, υπολογίζεται η μέση πληροφορία (εντροπία) για κάθε υποψήφιο συνδυασμό και επιλέγεται αυτός με την μέγιστη εντροπία. Σε περίπτωση ισοβαθμίας κάποιων συνδυασμών, επιλέγεται αυτός με την μικρότερη τιμή. Πιο αναλυτικά, στην αρχή του παιχνιδιού δεν παρέχεται κανένα στοιχείο στον παίκτη και ως αποτέλεσμα χρειάζεται να υπολογιστούν όλοι οι πιθανοί συνδυασμοί. Έτσι δημιουργείται ένα αρχικό σύνολο P_0. Αν συμβολίσουμε το σύνολο όλων των πιθανών αποτελεσμάτων ως R_i και τον συνδυασμό που επελέγη ως G_i για τον i γύρο ενός παιχνιδιού, τότε παρέχεται η πληροφορία (P_{i-1}, G_i, R_i), η οποία είναι και η πληροφορία που επιθυμούμε να μεγιστοποιήσουμε. Για χάρη απλότητας, ας ονομαστεί m_i. Όσο μεγαλύτερη γίνεται η ποσότητα της πληροφορίας που παρέχει το μήνυμα, το οποίο μπορούμε να το δούμε και σαν συνάρτηση $I(m_i)$, τόσο μικρότερο αναμένεται να γίνει το πλήθος των διαθέσιμων συνδυασμών. Συνεπώς, η πιθανότητα να παραχθεί το εκάστοτε μήνυμα, είναι ο λόγος:

$$Probability(m_i) = \frac{\#(Σετ\ πιθανών\ συνδυασμών\ που\ προκύπτει\ )}{\#(Αρχικό\ Σετ\ πιθανών\ συνδυασμών)}$$

και η ποσότητα της πληροφορίας που μπορεί να παραχθεί είναι:

$$I(m_i) = log_2\left(\frac{\#(Αρχικό\ σετ\ πιθανών\ συνδυασμών)}{\#(Σετ\ πιθανών\ συνδυασμών\ που\ προκύπτει\ )}\right)$$

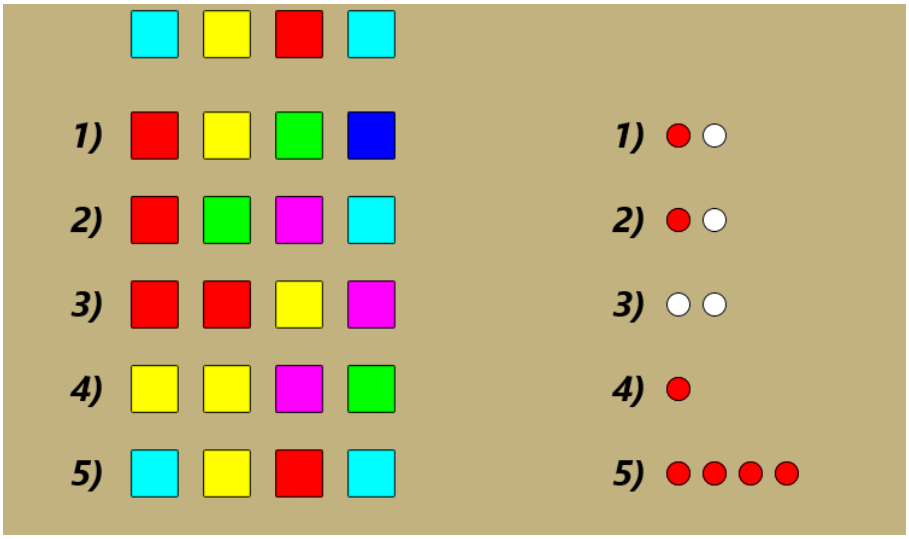
Αναλυτικά ως βήματα που ακολουθούνται είναι τα εξής:

1. Αρχικά δημιουργείται ένα σετ S από όλους τους πιθανούς κωδικούς.
2. Ως πρώτη δοκιμή, ξεκινάει με τον κωδικό «1234».
3. Λαμβάνει το αποτέλεσμα της δοκιμής και του κρυφού κωδικού.
4. Εάν η πληροφορία που λαμβάνει είναι τέσσερα κόκκινα σημάδια, το παιχνίδι έχει ολοκληρωθεί και ο αλγόριθμος τερματίζεται.
5. Διαφορετικά, διαγράφει οποιονδήποτε κωδικό, που δεν θα έδινε το ίδιο αποτέλεσμα, εφόσον η τρέχουσα δοκιμή ήταν ο κρυφός κωδικός.
6. Υπολογίζει για κάθε πιθανό κωδικό και κάθε πιθανό αποτέλεσμα που μπορεί να παραχθεί την μέση πληροφορία που μπορεί να προσφέρει.
7. Υπολογίζει την εντροπία *H* για κάθε πιθανό συνδυασμό και κάθε πιθανό αποτέλεσμα που μπορεί να παραχθεί, επιλέγοντας αυτόν που την μεγιστοποιεί.

$$H = \sum Probability(m_i) \times I(m_i)$$

8. Ο αλγόριθμος επαναλαμβάνεται από το βήμα 3.

Για την καλύτερη κατανόηση του αλγορίθμου ακολουθεί παράδειγμα, όπου ο κρυφός κωδικός είναι και πάλι ο «6216» και η χρήση των χρωμάτων επιτρέπεται παραπάνω από μία φορά.



Εικόνα 4.16: Maximum Entropy Algorithm – Βήματα Επίλυσης

Η πρώτη δοκιμή είναι η «1234» και αποτελεί πάντα την πρώτη δοκιμή. Αφού λάβει το αποτέλεσμα, ο πράκτορας αφαιρεί όλους τους κωδικούς εκείνους που δεν δίνουν το ίδιο αποτέλεσμα, εφόσον ο κρυφός κωδικός ήταν η δοκιμή που πραγματοποιήθηκε. Ύστερα και προκειμένου να πραγματοποιήσει την εκάστοτε δοκιμή, ο πράκτορας προσπαθεί να εντοπίσει ποιος κωδικός του επιστρέφει τη μεγαλύτερη εντροπία. Δηλαδή, υπολογίζεται η μέση πληροφορία (εντροπία) για κάθε υποψήφιο συνδυασμό και επιλέγεται αυτός με την μέγιστη.

Αξίζει να σημειωθεί πως ο λόγος που επιλέχτηκε ως πρώτη κίνηση ο κωδικός «1234» δεν είναι τυχαίος. Όπως και στην πορεία του αλγορίθμου, έτσι και στην αρχή του παιχνιδιού, σκοπός είναι να επιλέγεται ένας κωδικός που μεγιστοποιεί την εντροπία. Έτσι, με βάση τον Πίνακα 4.14, έγινε και η επιλογή της πρώτης προσπάθειας.

Πίνακας 4.14: Maximum Entropy Algorithm – Αποτελέσματα 1ης Δοκιμής

ΚΩΔΙΚΟΣ	**ΜΕΓΙΣΤΗ ΕΝΤΡΟΠΙΑ**
1111	1.498
1112	2.693
1122	2.885
1123	3.044
1234	**3.057**

Αντίστοιχα για τα επόμενα επίπεδα έχουν υπολογιστεί οι πρώτες δοκιμές.

Διαθέσιμα Χρώματα 8, Μήκος Κωδικού 4

Πίνακας 4.15: Maximum Entropy Algorithm – Αποτελέσματα 1ης Δοκιμής

ΚΩΔΙΚΟΣ	ΜΕΓΙΣΤΗ ΕΝΤΡΟΠΙΑ
1111	1.305
1112	2.402
1122	2.553
1123	2.795
1234	**2.899**

Επιλέγεται και πάλι ο κωδικός «1234», καθώς παρέχει την μέγιστη εντροπία.

Διαθέσιμα Χρώματα 6, Μήκος Κωδικού 5

Πίνακας 4.16: Maximum Entropy Algorithm – Αποτελέσματα 1ης Δοκιμής

ΚΩΔΙΚΟΣ	ΜΕΓΙΣΤΗ ΕΝΤΡΟΠΙΑ
11111	1.67
11112	2.89
11122	3.17
11123	3.25
11223	**3.35**
11234	3.29
12345	3.14

Η μέγιστη εντροπία επιτυγχάνεται με τον κωδικό «11223».

Διαθέσιμα Χρώματα 7, Μήκος Κωδικού 5

Πίνακας 4.17: Maximum Entropy Algorithm – Αποτελέσματα 1ης Δοκιμής

ΚΩΔΙΚΟΣ	**ΜΕΓΙΣΤΗ ΕΝΤΡΟΠΙΑ**
11111	1.56
11112	2.75
11122	3
11123	3.14
11223	3.25
11234	**3.26**
12345	3.18

Η μέγιστη εντροπία έχει επιτυγχάνεται με τον κωδικό «11234».

4.7 Serial Algorithm With Worst Case

Ο αλγόριθμος «Serial Algorithm With Worst Case» αποτελεί μία στρατηγική που προτείνεται στο πλαίσιο της εργασίας, με βάση την εμπειρία που αποκτήθηκε από την ενασχόληση με το παιχνίδι και τις υλοποιημένες στρατηγικές.

Ο συγκεκριμένος αλγόριθμος είναι μια προσπάθεια συνδυασμού των θετικών στοιχείων των αλγορίθμων «Serial Algorithm» και «Knuth Algorithm». Δηλαδή, πραγματοποιήθηκε προσπάθεια να κρατηθεί το χαμηλό κόστος μνήμης από τον πρώτο και η έξυπνη επιλογή κωδικών από τον δεύτερο. Το αρνητικό στοιχείο που εντοπίστηκε στην στρατηγική «Serial Algorithm», είναι το γεγονός ότι εάν έχει τοποθετηθεί κάποιο χρώμα το οποίο βρίσκεται στο τέλος της ιεραρχίας θα απαιτηθεί τουλάχιστον ίσος αριθμός προσπαθειών. Γι' αυτό τον λόγο, τώρα η ανάθεση των χρωμάτων θα γίνεται ανά δύο, δηλαδή πρώτα θα δοκιμαστεί ο κωδικός «1122», μετά ο «3344» και μετά ο «5566». Με αυτό τον τρόπο, θα χρειαστεί ο μισός αριθμός προσπαθειών συγκριτικά με πριν για να δοκιμαστούν όλα τα χρώματα. Όπως και πριν, έτσι και τώρα οι πιθανοί συνδυασμοί θα δημιουργούνται σταδιακά με βάση τα αποτελέσματα που παράγονται. Μόλις δημιουργηθούν όλοι οι πιθανοί συνδυασμοί, τότε έρχεται να λάβει μέρος η στρατηγική της χείριστης κατάστασης που έχει υλοποιηθεί στον «Knuth Algorithm», για την επιλογή της επόμενης δοκιμής. Με αυτόν τον τρόπο, επιχειρείται να συνδυαστούν οι δύο αλγόριθμοι. Στην περίπτωση που δοκιμαστούν όλα τα χρώματα και δεν έχουν σχηματιστεί πλήρως όλα τα πιθανά μοντέλα, τότε συμπληρώνονται ανάλογα με τα προηγούμενα αποτελέσματα.

Για να γίνει και πιο σαφές, παρουσιάζονται και τα βήματα του αλγορίθμου.

1. Πρώτα απ' όλα, ο αλγόριθμός αποθηκεύει στην μνήμη έναν κατάλογο με την διαθεσιμότητα των χρωμάτων. Δηλαδή ποια χρώματα μπορούν να μπουν σε ποιες θέσεις. Πέρα από τον προαναφερθέντα κατάλογο, ο αλγόριθμος αποθηκεύει στην μνήμη το μοντέλο «XXXX», όπου η τιμή X μπορεί να πάρει οποιαδήποτε τιμή των χρωμάτων. Έτσι, αναγκαστικά ο κωδικός που ψάχνει να βρει θα είναι της παραπάνω μορφής. Δηλαδή
    - $διαθέσιμα\ χρώματα\ = \{1, 2, 3, 4, 5, 6\}$
    - $X \in διαθέσιμα\ χρώματα$
    - $αρχικό\ μοντέλο\ = \ll XXXX \gg$
    - $κωδικός\ \in πιθανά\ μοντέλα$

2. Την πρώτη φορά θα δοκιμάσει τον κωδικό «1122».

3. Λαμβάνει το αποτέλεσμα της δοκιμής και του κρυφού κωδικού.

4. Εάν η πληροφορία που λαμβάνει είναι τέσσερα κόκκινα σημάδια, το παιχνίδι έχει ολοκληρωθεί και ο αλγόριθμος τερματίζεται.

5. Με βάση το αποτέλεσμα θα δημιουργηθούν τα νέα πιθανά μοντέλα. Επιπλέον, ανανεώνει την λίστα με την διαθεσιμότητα των χρωμάτων. Εφόσον για παράδειγμα έχουμε μόνο λευκά σημάδια για αποτέλεσμα, θα πρέπει να αφαιρεθούν τα αντίστοιχα χρώματα από τις αντίστοιχες θέσεις.

6. Η επόμενη δοκιμή θα περιέχει τα επόμενα δύο χρώματα.

7. Τα βήματα 3-6 επαναλαμβάνονται έως ότου δοκιμαστούν όλα τα χρώματα.

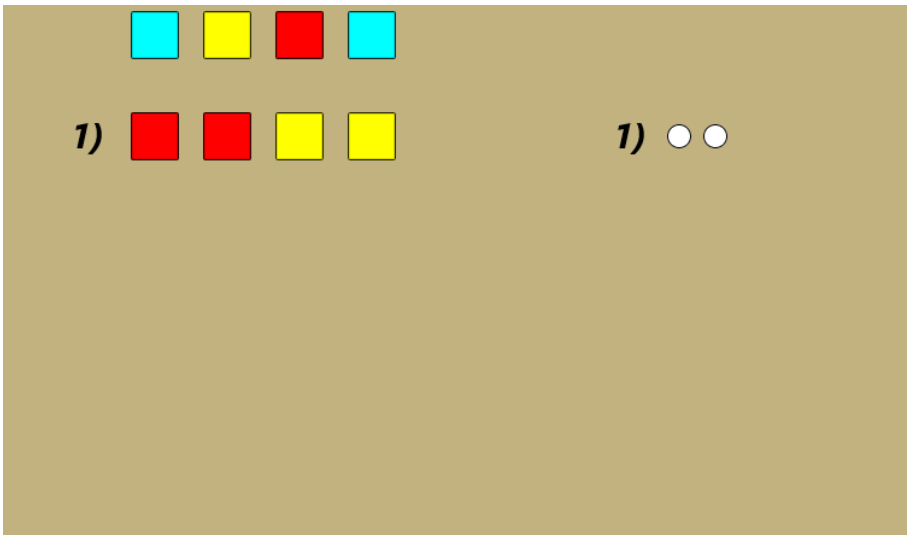
8. Ελέγχεται εάν όλα τα πιθανά μοντέλα έχουν σχηματιστεί. Εάν όχι τότε αυτό σημαίνει ότι κάποιο χρώμα υπάρχει παραπάνω από δύο φορές. Τότε, ανάλογα με τα προηγούμενα αποτελέσματα και ανάλογα με την διαθεσιμότητα των χρωμάτων, συμπληρώνονται ανάλογα οι κενές θέσεις των μοντέλων.

9. Αφού πλέον έχουν σχηματιστεί πλήρως όλοι οι ενδεχόμενοι κωδικοί, τότε η επιλογή της επόμενης δοκιμής γίνεται με βάση την ελαχιστοποίηση της χειρότερης κατάστασης.

10. Ο πράκτορας λαμβάνει το αποτέλεσμα της δοκιμής και του κρυφού κωδικού και αποκλείει όποιον κωδικό δεν θα παρήγαγε το ίδιο αποτέλεσμα, εφόσον η δοκιμή ήταν ο κωδικός που ψάχνουμε.

11. Τα βήματα 9 και 10 επαναλαμβάνονται έως ότου ολοκληρωθεί το παιχνίδι.

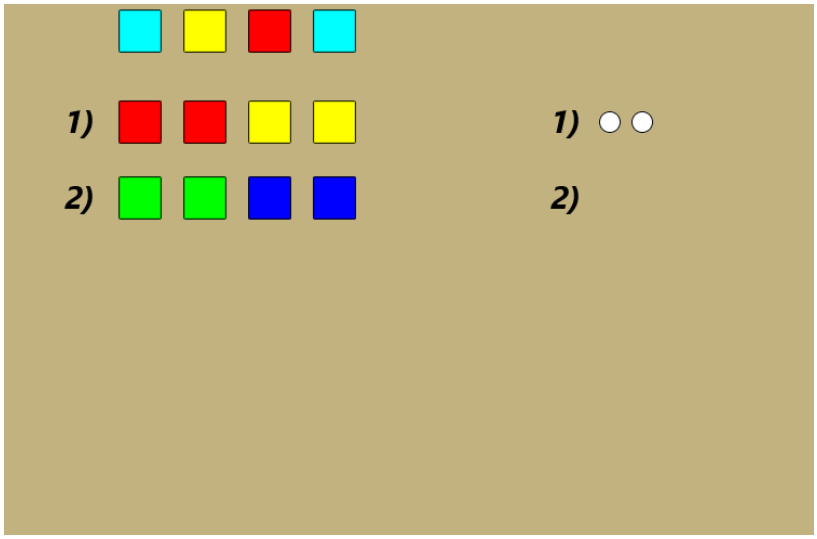
Παρακάτω παρουσιάζεται και ένα παράδειγμα για την ορθότερη κατανόηση. Ο κρυφός κωδικός είναι ο «5336» και η χρήση διπλότυπων επιτρέπεται.



Εικόνα 4.17: Serial Algorithm With Worst Case – Βήμα 1

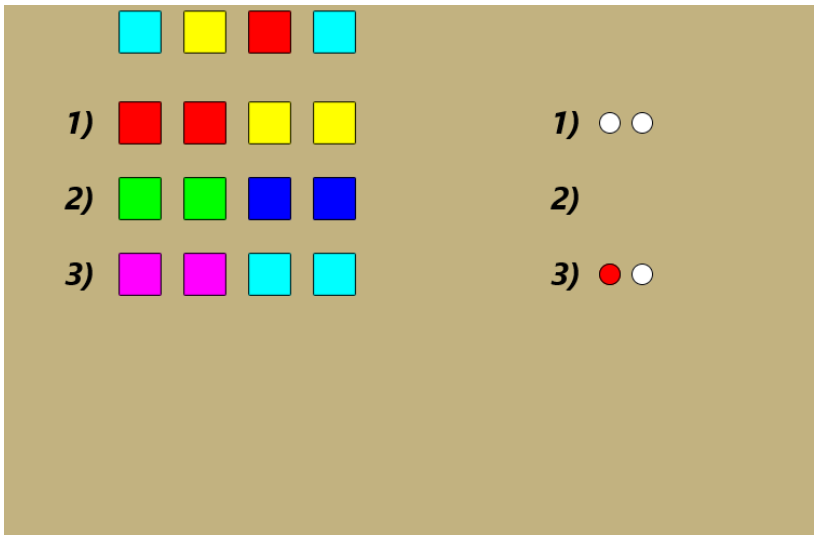
Η πρώτη δοκιμή όπως προαναφέρθηκε είναι η «1122». Σαν αποτέλεσμα εξάγει δύο λευκά σημάδια και ως εκ τούτου, ο πράκτορας αντιλαμβάνεται ότι το «Κόκκινο» ή/και το «Κίτρινο» υπάρχουν στον ζητούμενο κωδικό, σε διαφορετικές όμως θέσεις. Αυτό οδηγεί στην ανανέωση των πιθανών μοντέλων και έτσι τώρα υπάρχουν στην μνήμη τα εξής:

«22XX», «2X1X», «2XX1», «X21X», «X2X1», «XX11»



Εικόνα 4.18: Serial Algorithm With Worst Case – Βήμα 2

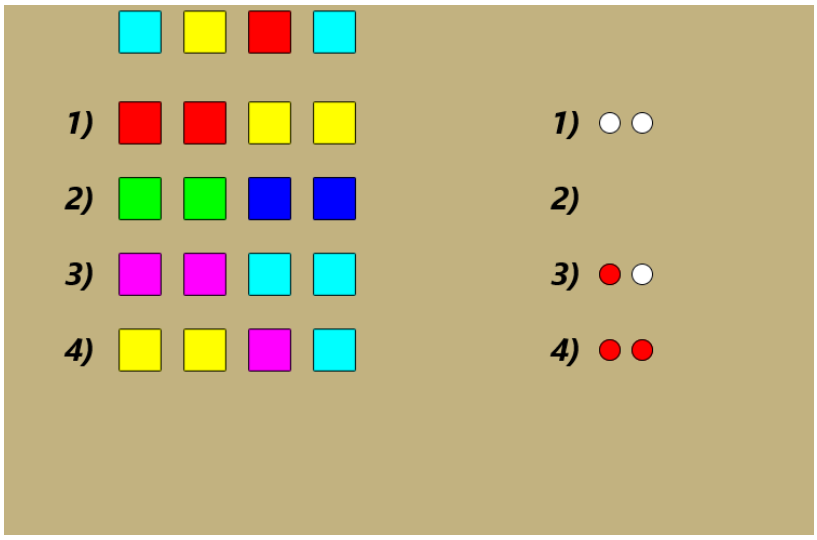
Έπειτα, θα δοκιμαστούν τα επόμενα δύο χρώματα, το «Πράσινο» και το «Μπλε» ή αλλιώς θα δοκιμαστεί ο κωδικός «3344». Με βάση το αποτέλεσμα που παράγεται, ο πράκτορας αντιλαμβάνεται ότι δεν υπάρχει κάποιο από τα δύο χρώματα και συνεπώς τα πιθανά μοντέλα παραμένουν αναλλοίωτα.



Εικόνα 4.19: Serial Algorithm With Worst Case – Βήμα 3

Η τελευταία δοκιμή που πραγματοποιείται με την ίδια λογική είναι ο κωδικός «5566», καθώς είναι και αυτός με τον οποίο ολοκληρώνεται η επαλήθευση όλων των χρωμάτων. Το γεγονός ότι αναγνωρίζονται δύο από τα τέσσερα χρώματα, σε συνδυασμό με το γεγονός ότι το ίδιο συνέβη και στην πρώτη δοκιμή, μας οδηγεί στο συμπέρασμα ότι όλα τα πιθανά μοντέλα θα συμπληρωθούν πλήρως και έχουν ως εξής:

«2256», «2265», «2515», «2551», «2616», «2661», «5215», «5251», «5611», «6216», «6261», «6511»

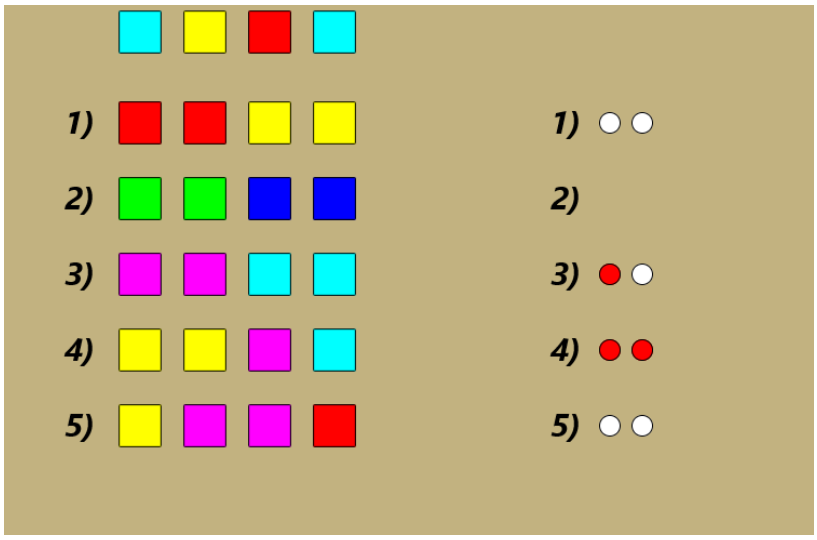


Εικόνα 4.20: Serial Algorithm With Worst Case – Βήμα 4

Αφού έχουν σχηματιστεί όλοι οι πιθανοί κωδικοί που μπορούν να τερματίσουν το παιχνίδι, εφαρμόζεται ο αλγόριθμος minimax του Knuth. Για κάθε πιθανό κωδικό,

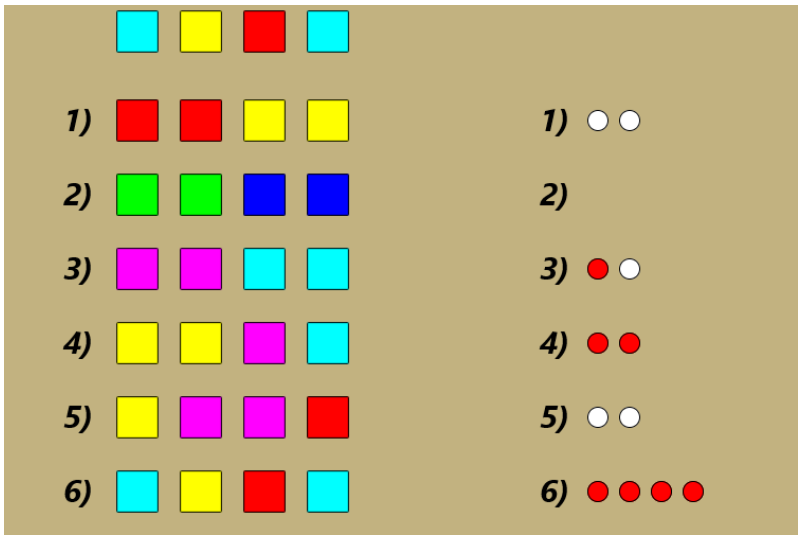
υπολογίζει πόσους κωδικούς μπορεί να εξαλείψει για κάθε πιθανό αποτέλεσμα. Για κάθε συνδυασμό κρατάει την χειρότερη περίπτωση και στο τέλος επιλέγει τον κωδικό με την μικρότερη χείριστη κατάσταση. Θα επιλεχθεί ο κωδικός με την καλύτερη χειρότερη κατάσταση. Έτσι προτιμάται ο κωδικός «2256», όπου με το αποτέλεσμα που λαμβάνει διατηρεί μόνο τους εξής κωδικούς:

«2551», «2616», «5251», «6216»



Εικόνα 4.21: Serial Algorithm With Worst Case – Βήμα 5

Αντίστοιχα με προηγουμένως, ο κωδικός που επιλέγεται να δοκιμαστεί είναι ο «2551». Ύστερα από το αποτέλεσμα που εξάγεται, θα απορριφθούν όσοι συνδυασμοί συγκριθούν με την δοκιμή και δεν δώσουν το ίδιο αποτέλεσμα. Έτσι ο μόνος κωδικός που ικανοποιεί όλες τις συνθήκες είναι ο «6216».



Εικόνα 4.22: Serial Algorithm With Worst Case – Βήμα 6

Τέλος, θα δοκιμαστεί ο μοναδικός συνδυασμός που έχει συγκρατήσει ο πράκτορας, που δεν είναι άλλος από τον «6216». Η λύση επιβεβαιώνεται, αφού σαν αποτέλεσμα έχουμε 4 κόκκινα σημάδια. Ο ζητούμενος κωδικός έχει βρεθεί με την χρήση έξι προσπαθειών.

5 ΑΠΟΤΕΛΕΣΜΑΤΑ

5.1 Συνδυαστική Αξιολόγηση Αλγορίθμων

Για να μπορέσουμε να πραγματοποιήσουμε σύγκριση αποτελεσμάτων, οι αλγόριθμοι δοκιμάστηκαν για κάθε πιθανό συνδυασμό που μπορεί να προκύψει σε κάθε ένα από τα επίπεδα. Οι αλγόριθμοι «Serial Algorithm», «Knuth Algorithm», «Expected Size Algorithm», «Maximum Entropy Algorithm» και «Serial Algorithm With Worst Case» είναι ντετερμινιστικοί και ως εκ τούτου ο κάθε πιθανός κωδικός αρκεί να δοκιμαστεί μία μόνο φορά, καθώς πάντοτε θα δίνει τα ίδια αποτελέσματα. Αντίθετα ο αλγόριθμος «Simple Algorithm», που είναι μη ντετερμινιστικός και κάθε φορά παράγει τυχαία αποτελέσματα, κάθε κωδικός δοκιμάστηκε 100 φορές, με σκοπό η τυχαιότητα που χαρακτηρίζει τον αλγόριθμο να εξισορροπηθεί.

Όσον αφορά την δυσκολία του κάθε επιπέδου, έχει να κάνει με το πλήθος των κωδικών που δημιουργούνται. Όσο αυξάνονται οι διαθέσιμοι κωδικοί, είναι λογικό να αυξάνεται και ο αριθμός των δοκιμών που απαιτούνται για να εντοπιστεί ο κρυφός κωδικός. Ταυτόχρονα οι αλγόριθμοι που χρησιμοποιούν ευρετική συνάρτηση (Knuth, Expected Size, Maximum Entropy) έχουν γραμμική πολυπλοκότητα ως προς το πλήθος των πιθανών συνδυασμών, με αποτέλεσμα να αυξάνεται σημαντικά και ο χρόνος που απαιτείται για την ολοκλήρωση του παιχνιδιού.

Πιο αναλυτικά, όλοι οι συνδυασμοί χρωμάτων που δημιουργούνται για πλήθος διαθέσιμων χρωμάτων C και μέγεθος κωδικού p είναι C^p, καθώς αποτελεί διάταξη με επανάληψη. Ενώ όταν η χρήση διπλότυπων χρωμάτων απαγορεύεται, υφίσταται διάταξη των C στοιχείων ανά p, δηλαδή δημιουργούνται $\frac{C!}{(C-p)!}$ συνδυασμοί. Συγκεκριμένα για τις ανάγκες της εργασίας έχουν υλοποιηθεί 4 διαφορετικές περιπτώσεις, όσον αφορά την διαθεσιμότητα των χρωμάτων και το μέγεθος του κωδικού. Η κάθε περίπτωση μπορεί να λειτουργήσει είτε με την πολλαπλή χρήση των χρωμάτων είτε όχι. Έτσι έχουν κατασκευαστεί 8 πίστες, προκειμένου να δοκιμαστούν και να αξιολογηθούν οι πράκτορες.

1. *6 διαθέσιμα χρώματα και μήκος κωδικού 4*
    a. Πολλαπλή χρήση χρωμάτων: $6^4 = 1296$ συνδυασμοί
    b. Κάθε χρώμα χρησιμοποιείται μία φορά: $\frac{6!}{(6-4)!} = 360\ συνδυασμοί$
2. *8 διαθέσιμα χρώματα και μήκος κωδικού 4*
    a. Πολλαπλή χρήση χρωμάτων: $8^4 = 4096$ συνδυασμοί
    b. Κάθε χρώμα χρησιμοποιείται μία φορά: $\frac{8!}{(8-4)!} = 1680\ συνδυασμοί$
3. *6 διαθέσιμα χρώματα και μήκος κωδικού 5*
    a. Πολλαπλή χρήση χρωμάτων: $6^5 = 7776$ συνδυασμοί
    b. Κάθε χρώμα χρησιμοποιείται μία φορά: $\frac{6!}{(6-5)!} = 720\ συνδυασμοί$
4. *7 διαθέσιμα χρώματα και μήκος κωδικού 5*
    a. Πολλαπλή χρήση χρωμάτων: $7^5 = 16807$ συνδυασμοί
    b. Κάθε χρώμα χρησιμοποιείται μία φορά: $\frac{7!}{(7-5)!} = 2520\ συνδυασμοί$

Τέλος, προκειμένου να εξάγουμε ορθά συμπεράσματα για την απόδοση του κάθε αλγορίθμου, έχουν υπολογιστεί ο ελάχιστος, ο μέγιστος, ο μέσος όρος και η τυπική απόκλιση του χρόνου και των προσπαθειών που απαιτούνται για την ολοκλήρωση του παιχνιδιού. Μια χαμηλή τυπική απόκλιση υποδηλώνει ότι τα σημεία των δεδομένων τείνουν να είναι κοντά στο μέσο όρο του συνόλου, ενώ μία υψηλή τυπική απόκλιση υποδεικνύει ότι τα στοιχεία απλώνονται πάνω από ένα ευρύτερο φάσμα των τιμών. Όπως θα παρατηρηθεί και αργότερα, ο ελάχιστος αριθμός είτε του χρόνου είτε των προσπαθειών, δεν είναι εξίσου σημαντικός με τον μέσο όρο ή το μέγιστο αριθμό, καθώς σε όλους τους αλγορίθμους, εκτός από τον «Simple Algorithm», η πρώτη δοκιμή είναι πάντα η ίδια. Έτσι θα υπάρχει πάντα ένας κωδικός που θα εντοπίζεται με την πρώτη προσπάθεια και σε πολύ σύντομο χρονικό διάστημα. Αντίστοιχα, το πλήθος των επαναλήψεων που έχουν διεξαχθεί για τον «Simple Algorithm», αφήνουν ανοικτό το ενδεχόμενο και μάλιστα με μεγάλη πιθανότητα, να συναντήσουμε το ίδιο φαινόμενο.

5.2 Απόδοση Αλγορίθμων

5.2.1 Serial Algorithm

Πίνακας 5.1: Αποτελέσματα Χρόνου «Serial Algorithm»

Serial Algorithm							
Level				**Time(ms)**			
Συνδυασμοί	Available Colors	Code Length	Duplicates	Minimum	Maximum	Average	Standard Deviation
1296	6	4	✔	0	7	0.08	0.33
360	6	4	✗	0	9	0.80	0.84
4096	8	4	✔	0	14	0.07	0.35
1680	8	4	✗	0	10	1.33	1.41
7776	6	5	✔	0	12	0.15	0.42
720	6	5	✗	0	26	11.48	7.07
16807	7	5	✔	0	13	0.13	0.42
2520	7	5	✗	0	129	28.04	34.55

Πίνακας 5.2: Αποτελέσματα Προσπαθειών «Serial Algorithm»

Serial Algorithm							
Level				**Attempts**			
Συνδυασμοί	Available Colors	Code Length	Duplicates	Minimum	Maximum	Average	Standard Deviation
1296	6	4	✔	1	9	5.76	1.05
360	6	4	✗	1	6	4.15	0.86
4096	8	4	✔	1	11	7.21	1.29
1680	8	4	✗	1	7	4.90	0.96
7776	6	5	✔	1	11	6.22	1.03
720	6	5	✗	1	8	5.04	1.07
16807	7	5	✔	1	12	6.90	1.09
2520	7	5	✗	1	8	5.16	0.98

Όπως μπορούμε να διαπιστώσουμε και από τον Πίνακα 5.1, ο Serial Algorithm είναι αρκετά αποτελεσματικός όσον αφορά το χρονικό κομμάτι του υπολογισμού. Το γεγονός ότι δεν χρησιμοποιεί ευρετική συνάρτηση για την επιλογή της επόμενης δοκιμής, αλλά ούτε και χρειάζεται να υπολογίσει όλους τους κωδικούς, καθώς τους διαμορφώνει σταδιακά, του προσδίδει ταχύτητα και άμεση εύρεση του κωδικού. Από την άλλη πλευρά, παρουσιάζει κάποια αστάθεια στο σύνολο των προσπαθειών. Όπως μπορούμε να παρατηρήσουμε στον Πίνακα 5.2, ο μέσος όρος των προσπαθειών, παρουσιάζει μεγάλες διαφορές σε σχέση με τον μέγιστο αριθμό προσπαθειών που χρειάστηκε για τον εντοπισμό του χειρότερου σεναρίου σε κάθε πίστα. Αυτό συμβαίνει, διότι στα χρώματα έχει αποδοθεί μια ιεραρχική λίστα, προκειμένου να είναι πιο εύκολη η επίλυση του παιχνιδιού από τον πράκτορα. Έτσι, προκειμένου ο αλγόριθμος να εντοπίσει έναν κωδικό που περιέχει κάποιο χρώμα που βρίσκεται τελευταίο ιεραρχικά, θα χρειαστεί τουλάχιστον ίδιο αριθμό ευκαιριών. Σε ορισμένες περιπτώσεις ενδέχεται να χρειαστεί και παραπάνω, ανάλογα με το πλήθος των πιθανών μοντέλων που έχουν σχηματιστεί. Αντίστοιχα ένας κωδικός που εμπεριέχει χρώματα που είναι στην αρχή της ιεραρχικής λίστας, θα εντοπίζεται πιο άμεσα. Παρόλα αυτά, αν παρατηρήσουμε την τυπική απόκλιση, η οποία κυμαίνεται σε χαμηλά επίπεδα, οι περισσότεροι κωδικοί εντοπίζονται κοντά στον μέσο όρο.

5.2.2 Simple Algorithm

Πίνακας 5.3: Αποτελέσματα Χρόνου «Simple Algorithm»

Simple Algorithm							
Level				Time(ms)			
Συνδυασμοί	Available Colors	Code Length	Duplicates	Minimum	Maximum	Average	Standard Deviation
1296	6	4	✔	0	15	1.25	0.50
360	6	4	✗	0	12	0.44	0.62
4096	8	4	✔	2	31	3.79	0.58
1680	8	4	✗	0	23	1.95	0.45
7776	6	5	✔	14	1862	18.71	4.93
720	6	5	✗	0	9	0.91	0.49
16807	7	5	✔	32	2473	41.83	4.80
2520	7	5	✗	5	41	7.21	0.73

Πίνακας 5.4: Αποτελέσματα Προσπαθειών «Simple Algorithm»

Simple Algorithm							
Level				Attempts			
Συνδυασμοί	Available Colors	Code Length	Duplicates	Minimum	Maximum	Average	Standard Deviation
1296	6	4	✔	1	8	4.64	0.88
360	6	4	✗	1	7	4.13	0.84
4096	8	4	✔	1	10	5.43	1.02
1680	8	4	✗	1	8	4.86	0.93
7776	6	5	✔	1	8	5.07	0.84
720	6	5	✗	1	8	4.94	1.00
16807	7	5	✔	1	9	5.48	0.89
2520	7	5	✗	1	8	5.08	0.95

Όπως και «Serial Algorithm», έτσι και ο «Simple Algorithm» παρουσιάζει αρκετά ικανοποιητικά αποτελέσματα όσον αφορά το χρονικό κομμάτι της εύρεσης του κωδικού. Αυτό οφείλεται στο γεγονός, ότι δεν χρησιμοποιεί κάποια ευρετική συνάρτηση για τον εντοπισμό της επόμενης δοκιμής, αλλά την παράγει τυχαία. Αντίστοιχα παρατηρούμε ότι υπάρχει μία συνέπεια στον αριθμό τον προσπαθειών που απαιτούνται για την επίλυση του παιχνιδιού, αφού τόσο οι μέσοι όροι, όσο και οι χειρότερες περιπτώσεις, κυμαίνονται στα ίδια επίπεδα. Στα αρνητικά του συγκεκριμένου αλγορίθμου, πρέπει να προσθέσουμε ότι είναι μη ντετερμινιστικός, δηλαδή όπως προαναφέρθηκε παράγει τυχαία τους κωδικούς. Αυτό έχει σαν αποτέλεσμα, να μην εγγυάται την εξαγωγή των ίδιων αποτελεσμάτων για τον ίδιο κωδικό κάθε φορά.

5.2.3 Knuth Algorithm

Πίνακας 5.5: Αποτελέσματα Χρόνου «Knuth Algorithm»

Knuth Algorithm							
Level				Time(ms)			
Συνδυασμοί	Available Colors	Code Length	Duplicates	Minimum	Maximum	Average	Standard Deviation
1296	6	4	✔	3	23443	14034.60	6881.59
360	6	4	✗	2	4363	1196.90	1665.74
4096	8	4	✔	4	4066	2418.78	1103.85
1680	8	4	✗	2	4299	1613.83	1721.13
7776	6	5	✔	17	16263	9790.02	4593.06
720	6	5	✗	9	396	238.94	107.85
16807	7	5	✔	19	38546	18826.46	7812.47
2520	7	5	✗	12	23458	14156.18	5156.16

Πίνακας 5.6: Αποτελέσματα Προσπαθειών «Knuth Algorithm»

Knuth Algorithm							
Level				Attempts			
Συνδυασμοί	Available Colors	Code Length	Duplicates	Minimum	Maximum	Average	Standard Deviation
1296	6	4	✔	1	5	4.48	0.62
360	6	4	✗	1	7	4.36	0.92
4096	8	4	✔	1	7	5.52	0.63
1680	8	4	✗	1	7	4.78	0.85
7776	6	5	✔	1	6	5.20	0.57
720	6	5	✗	1	6	5.31	0.73
16807	7	5	✔	1	7	5.35	0.68
2520	7	5	✗	1	7	5.22	0.84

Ο «Knuth Algorithm» παρουσιάζει αρκετά ικανοποιητικά αποτελέσματα στο κομμάτι των προσπαθειών, γεγονός που επιβεβαιώνεται και από τα αποτελέσματα του Πίνακα 5.6., καθώς σε κανένα επίπεδο του παιχνιδιού δεν ξεπερνά τις 7 προσπάθειες. Επιπρόσθετα, αν εστιάσει κανείς στον μέσο όρο και στην τυπική απόκλιση, αντιλαμβάνεται ότι είναι αρκετά αξιόπιστος στο κομμάτι των συνολικών προσπαθειών. Στον αντίποδα, είναι αρκετά απαιτητικός όσον αφορά το χρονικό κομμάτι. Γι' αυτό τον λόγο και προκειμένου να μπορούν να παραχθούν τα αποτελέσματα σε εύλογο χρονικό διάστημα, κρίθηκε αναγκαία η χρήση πολυνηματικού προγραμματισμού (multithreading) από το επίπεδο ήδη των 8 διαθέσιμων χρωμάτων και κωδικού πλήθος 4. Η πολυνηματικότητα είναι η ικανότητα μίας κεντρικής μονάδας επεξεργασίας (CPU) να παρέχει πολλαπλά νήματα εκτέλεσης ταυτόχρονα. Έτσι, επιδιώκει να αυξήσει την χρήση του πυρήνα χρησιμοποιώντας παραλληλισμό επιπέδου νήματος, καθώς και παραλληλισμό σε επίπεδο εντολών. Με αυτόν τον τρόπο επιτυγχάνεται μία βελτίωση στον χρόνο που απαιτείται για να υπολογιστεί η εκάστοτε δοκιμή. Παρ' όλη όμως την προσπάθεια που πραγματοποιήθηκε για την πιο άμεση εξαγωγή του παιχνιδιού, ο χρόνος που απαιτείται για τον υπολογισμό των δοκιμών είναι αρκετός, ιδιαίτερα στο τελευταίο επίπεδο των 7 διαθέσιμων χρωμάτων και για πλήθος κωδικού 5, όπου όλοι οι πιθανοί συνδυασμοί είναι 16807 και ο μέσος όρος εντοπισμού του εκάστοτε κωδικού είναι 18,83 δευτερόλεπτα. Στο χρονικό κομμάτι του υπολογισμού, ο «Knuth Algorithm» είναι ο πιο απαιτητικός αλγόριθμος.

5.2.4 Expected Size Algorithm

Πίνακας 5.7: Αποτελέσματα Χρόνου «Expected Size Algorithm»

Expected Size Algorithm							
Level				Time(ms)			
Συνδυασμοί	Available Colors	Code Length	Duplicates	Minimum	Maximum	Average	Standard Deviation
1296	6	4	✔	1	310	158.24	98.74
360	6	4	✗	0	34	20.08	11.02
4096	8	4	✔	2	4277	2139.28	1423.50
1680	8	4	✗	1	1065	434.25	384.62
7776	6	5	✔	15	2558	1090.53	757.02
720	6	5	✗	2	127	63.66	39.52
16807	7	5	✔	38	15928	6207.80	4315.05
2520	7	5	✗	9	2318	1151.96	673.84

Πίνακας 5.8: Αποτελέσματα Προσπαθειών «Expected Size Algorithm»

Expected Size Algorithm							
Level				Attempts			
Συνδυασμοί	Available Colors	Code Length	Duplicates	Minimum	Maximum	Average	Standard Deviation
1296	6	4	✔	1	8	5.04	1.02
360	6	4	✗	1	6	4.15	0.86
4096	8	4	✔	1	11	6.26	1.35
1680	8	4	✗	1	7	4.90	0.96
7776	6	5	✔	1	8	5.02	0.83
720	6	5	✗	1	8	4.99	1.04
16807	7	5	✔	1	9	5.42	0.86
2520	7	5	✗	1	8	5.13	0.96

Όπως και με τον «Knuth Algorithm», έτσι και ο «Expected Size Algorithm» διαθέτει ευρετική συνάρτηση για τον εντοπισμό της εκάστοτε δοκιμής. Το γεγονός αυτό προσδίδει χρονική καθυστέρηση στον αλγόριθμο γεγονός που επιβεβαιώνεται και από τα αποτελέσματα του Πίνακα 5.7. Όπως και προηγουμένως, έτσι και στην παρούσα στρατηγική, κρίθηκε αναγκαίο να γίνει χρήση πολυνηματικού προγραμματισμού, προκειμένου η εξαγωγή των αποτελεσμάτων να μην λειτουργήσει ως τροχοπέδη στην εκπόνηση της διπλωματικής εργασίας. Από το στάδιο λοιπόν που αυξάνεται το μέγεθος του κωδικού και γίνεται 5, έγινε χρήση πολυνηματικού προγραμματισμού. Όσον αφορά το κομμάτι των συνολικών προσπαθειών, ο αλγόριθμος φαίνεται να ανταπεξέρχεται ικανοποιητικά, καθώς ο μέσος όρος σε όλα τα επίπεδα κυμαίνεται σε χαμηλά επίπεδα, όπως και η τυπική απόκλιση. Εξαίρεση αποτελεί το επίπεδο των 8 διαθέσιμων χρωμάτων και μέγεθος κωδικού 4, όπου για την χειρότερη περίπτωση χρειάζεται διψήφιος αριθμός δοκιμών.

5.2.5 Maximum Entropy Algorithm

Πίνακας 5.9: Αποτελέσματα Χρόνου «Maximum Entropy Algorithm»

Max Entropy Algorithm							
Level				Time(ms)			
Συνδυασμοί	Available Colors	Code Length	Duplicates	Minimum	Maximum	Average	Standard Deviation
1296	6	4	✔	0	341	146.44	117.77
360	6	4	✗	0	34	19.80	11.05
4096	8	4	✔	2	3473	1853.99	1280.68
1680	8	4	✗	1	3598	443.42	400.92
7776	6	5	✔	16	2599	1131.61	786.95
720	6	5	✗	2	132	64.47	40.37
16807	7	5	✔	37	15638	6408.76	4801.02
2520	7	5	✗	9	2388	1197.49	702.79

Πίνακας 5.10: Αποτελέσματα Προσπαθειών «Maximum Entropy Algorithm»

Max Entropy Algorithm							
Level				Attempts			
Συνδυασμοί	Available Colors	Code Length	Duplicates	Minimum	Maximum	Average	Standard Deviation
1296	6	4	✔	1	6	4.47	0.75
360	6	4	✗	1	6	4.04	0.77
4096	8	4	✔	1	8	5.13	0.85
1680	8	4	✗	1	7	4.74	0.84
7776	6	5	✔	1	7	4.80	0.69
720	6	5	✗	1	8	4.81	0.91
16807	7	5	✔	1	8	5.20	0.74
2520	7	5	✗	1	7	4.95	0.84

Αντίστοιχα με τους δύο προαναφερθέντες αλγορίθμους και η παρούσα στρατηγική, διαθέτει ευρετική συνάρτηση. Για τον λόγο αυτό και για τον πιο άμεσο υπολογισμό των αποτελεσμάτων χρειάστηκε να χρησιμοποιηθεί πολυνηματικός προγραμματισμός από το επίπεδο που αυξάνεται το μέγεθος του κρυφού κωδικού και από 4 γίνεται 5. Όπως αποδεικνύουν οι μέσοι χρόνοι του Πίνακα 5.9, η διαδικασία αυτή ήταν επιτυχημένη, καθώς δεν απαιτείται παραπάνω από 7 δευτερόλεπτα για την εύρεση του κρυφού κωδικού. Αντίστοιχα, παρατηρούμε ότι ο αλγόριθμος είναι αρκετά αποτελεσματικός, αφού σε καμία περίπτωση δεν χρειάζεται παραπάνω από 8 προσπάθειες, προκειμένου να εντοπίσει τον κρυφό κωδικό.

5.2.6 Serial With Worst Case Algorithm

Πίνακας 5.11: Αποτελέσματα Χρόνου «Serial With Worst Case Algorithm»

Serial With Worst Case Algorithm							
Level				Time(ms)			
Συνδυασμοί	Available Colors	Code Length	Duplicates	Minimum	Maximum	Average	Standard Deviation
1296	6	4	✔	0	44	2.37	5.50
360	6	4	✗	0	368	89.40	111.50
4096	8	4	✔	0	46	2.17	4.37
1680	8	4	✗	0	368	105.73	119.50
7776	6	5	✔	0	6037	112.43	440.10
720	6	5	✗	0	27	11.27	6.79
16807	7	5	✔	0	56543	2038.65	5615.22
2520	7	5	✗	0	133	28.98	35.81

Πίνακας 5.12: Αποτελέσματα Προσπαθειών «Serial With Worst Case Algorithm»

Serial With Worst Case Algorithm							
Level				Attempts			
Συνδυασμοί	Available Colors	Code Length	Duplicates	Minimum	Maximum	Average	Standard Deviation
1296	6	4	✔	1	8	5.50	0.87
360	6	4	✗	1	7	5.07	0.85
4096	8	4	✔	1	9	6.63	0.88
1680	8	4	✗	1	7	5.31	0.84
7776	6	5	✔	1	8	5.95	0.78
720	6	5	✗	1	8	5.04	1.07
16807	7	5	✔	1	10	6.46	0.99
2520	7	5	✗	1	8	5.16	0.98

Όπως μπορούμε εύκολα να παρατηρήσουμε ο συνδυασμός του «Serial Algorithm» και του «Knuth Algorithm» προσδίδει χρονική καθυστέρηση στον εντοπισμό της λύσης. Η παρατήρηση αυτή είναι απόλυτα λογική, καθώς πλέον δεν έχει αλλάξει μόνο ο τρόπος της σειριακής ανάθεσης των κωδικών, αλλά επιπλέον ο αλγόριθμος αξιοποιεί την ευρετική συνάρτηση της ελαχιστοποίησης στην χείριστη κατάσταση. Όσον αφορά την απόδοση του αλγορίθμου παρατηρούμε ότι κυμαίνεται στα ίδια επίπεδα με τον «Serial Algorithm».

5.3 Συγκριτική Αξιολόγηση Αλγορίθμων

Πίνακας 5.13: Μέσος Χρόνος Προσπαθειών Για Κωδικούς 4 Χρωμάτων

Μέσος Χρόνος Προσπαθειών (ms)				
Πράκτορες	**Μέγεθος Κωδικού: 4**			
	Διαθέσιμα Χρώματα: 6		**Διαθέσιμα Χρώματα: 8**	
	Διπλότυπα: Ναι	**Διπλότυπα: Όχι**	**Διπλότυπα: Ναι**	**Διπλότυπα: Όχι**
Serial	**0.08**	0.80	**0.07**	**1.33**
Simple	1.25	**0.44**	3.79	1.95
Knuth	14034.60	1196.90	2418.78	1613.83
Expected Size	158.24	20.08	2139.28	434.25
Maximum Entropy	146.44	19.80	1853.99	443.42
Serial Worst Case	2.37	89.40	2.17	105.73
Βέλτιστος	**Serial**	**Simple**	**Serial**	**Serial**

Πίνακας 5.14: Μέσος Χρόνος Προσπαθειών Για Κωδικούς 5 Χρωμάτων

Μέσος Χρόνος Προσπαθειών (ms)				
Πράκτορες	**Μέγεθος Κωδικού: 5**			
	Διαθέσιμα Χρώματα: 6		**Διαθέσιμα Χρώματα: 7**	
	Διπλότυπα: Ναι	**Διπλότυπα: Όχι**	**Διπλότυπα: Ναι**	**Διπλότυπα: Όχι**
Serial	**0.15**	11.48	**0.13**	28.04
Simple	18.71	**0.91**	41.83	**7.21**
Knuth	9790.02	238.94	18826.46	14156.18
Expected Size	1090.53	63.66	6207.80	1151.96
Maximum Entropy	1131.61	64.47	6408.76	1197.49
Serial Worst Case	112.43	11.27	2038.65	28.98
Βέλτιστος	**Serial**	**Simple**	**Serial**	**Simple**

Πίνακας 5.15: Μέσος Όρος Προσπαθειών Για Κωδικούς 4 Χρωμάτων

Μέσος Όρος Προσπαθειών				
Πράκτορες	**Μέγεθος Κωδικού: 4**			
	Διαθέσιμα Χρώματα: 6		**Διαθέσιμα Χρώματα: 8**	
	Διπλότυπα: Ναι	**Διπλότυπα: Όχι**	**Διπλότυπα: Ναι**	**Διπλότυπα: Όχι**
Serial	5.76	4.15	7.21	4.90
Simple	4.64	4.13	5.43	4.86
Knuth	4.48	4.36	5.52	4.78
Expected Size	5.04	4.15	6.26	4.90
Maximum Entropy	**4.47**	**4.04**	**5.13**	**4.74**
Serial Worst Case	5.50	5.07	6.63	5.31
Βέλτιστος	**Maximum Entropy**	**Maximum Entropy**	**Maximum Entropy**	**Maximum Entropy**

Πίνακας 5.16: Μέσος Όρος Προσπαθειών Για Κωδικούς 5 Χρωμάτων

Μέσος Όρος Προσπαθειών				
Πράκτορες	**Μέγεθος Κωδικού: 5**			
	Διαθέσιμα Χρώματα: 6		**Διαθέσιμα Χρώματα: 7**	
	Διπλότυπα: Ναι	**Διπλότυπα: Όχι**	**Διπλότυπα: Ναι**	**Διπλότυπα: Όχι**
Serial	6.22	5.04	6.90	5.16
Simple	5.07	4.94	5.48	5.08
Knuth	5.20	5.31	5.35	5.22
Expected Size	5.02	4.99	5.42	5.13
Maximum Entropy	**4.80**	**4.81**	**5.20**	**4.95**
Serial Worst Case	5.95	5.04	6.46	5.16
Βέλτιστος	**Maximum Entropy**	**Maximum Entropy**	**Maximum Entropy**	**Maximum Entropy**

Όπως μπορούμε να διαπιστώσουμε από τους μέσους όρους του Πίνακα 5.13 και Πίνακα 5.14, ο «Serial Algorithm» είναι ιδιαίτερα γρήγορος αλγόριθμος, ειδικά όταν η χρήση των χρωμάτων επιτρέπεται παραπάνω από μία φορά. Όταν τα χρώματα μπορούν να τοποθετηθούν μόνο μία φορά φαίνεται να λειτουργεί καλύτερα ο «Simple Algorithm». Τα αποτελέσματα αυτά είναι αρκετά λογικά, καθώς οι αλγόριθμοι που δεν αξιοποιούν κάποια ευρετική συνάρτηση για να πραγματοποιήσουν τις δοκιμές τους, πραγματοποιούν ταυτόχρονα και λιγότερους υπολογισμούς. Ο «Serial Algorithm» παρατηρούμε ότι στις περιπτώσεις που τα διπλότυπα χρώματα επιτρέπονται ανταποκρίνεται καλύτερα, παρόλο που θεωρητικά είναι πιο δύσκολο να επιλύσει κανείς αυτές τις πίστες. Το κύριο όμως χαρακτηριστικό του αλγορίθμου είναι το γεγονός ότι αρκεί μόνο μία δοκιμή, προκειμένου να καταλάβει εάν ένα χρώμα υπάρχει και πόσες φορές βρίσκεται στον κωδικό. Από την άλλη πλευρά, το γεγονός ότι ο «Simple Algorithm» παράγει τυχαία αποτελέσματα, μας είχε προϊδεάσει για την γρήγορη αποτελεσματικότητά του, κάτι που επιβεβαιώνεται και από τους παραπάνω πίνακες. Όσον αφορά την απόδοση της κάθε στρατηγικής στον συνολικό αριθμό προσπαθειών, ο βέλτιστος αλγόριθμος, όπως αποδεικνύουν και οι μέσοι όροι των πινάκων 5.15 και 5.16, είναι ο «Maximum Entropy Algorithm». Παρά το γεγονός ότι ο «Knuth Algorithm» μπορεί να εντοπίσει την λύση το πολύ σε 5 προσπάθειες (κλασσική έκδοση παιχνιδιού) δεν τον καθιστά κατ' ανάγκη καλύτερο, καθώς στην σύγκριση των μέσων προσπαθειών, αν και οριακά, παρουσιάζει χειρότερα αποτελέσματα. Ολοκληρώνοντας την σύγκριση των αποτελεσμάτων, ο αλγόριθμος που εν τέλει καταφέρνει να συνδυάσει και την αποτελεσματικότητα και να παραγάγει άμεσα αποτελέσματα, είναι ο «Simple Algorithm».

6 ΣΥΜΠΕΡΑΣΜΑΤΑ

6.1 Σχολιασμός

Όπως διαπιστώσαμε και στο προηγούμενο κεφάλαιο, όλοι οι πράκτορες παράγουν ικανοποιητικά αποτελέσματα, τόσο στο αρχικό μοντέλο του παιχνιδιού, όσο και στις υπόλοιπες περιπτώσεις που δημιουργήθηκαν. Αρκετές από τις στρατηγικές που αναπτύχθηκαν είναι ντετερμινιστικές, γεγονός που σημαίνει ότι θα μπορούσαμε να έχουμε προ-υπολογίσει όλες τις περιπτώσεις, προκειμένου οι πράκτορες να λειτουργούν αρκετά γρήγορα. Η τεχνική αυτή δεν πραγματοποιήθηκε, καθώς προτιμήθηκε να μελετήσουμε τον κάθε πράκτορα, σαν ένα πράκτορα που προσπαθεί να επιλύσει το παιχνίδι επί τόπου. Η αξίωση αυτή, όπως είναι λογικό έκανε πιο αργούς τους πράκτορές μας και γι' αυτό, καθοριστικό ρόλο στην εργασία έπαιξε η χρήση πολυνηματικού προγραμματισμού. Χωρίς αυτήν, η εξαγωγή των αποτελεσμάτων θα ήταν αρκετά χρονοβόρα και θα καθυστερούσε σε μεγάλο βαθμό την υλοποίηση της παρούσας εργασίας. Αξιοποιήθηκε μόνο στις περιπτώσεις που η στρατηγική που ακολουθείται χρησιμοποιεί ευρετική συνάρτηση, καθώς προσθέτει χρονική καθυστέρηση λόγω των υπολογισμών.

6.2 Μελλοντικές Προσθήκες

Σίγουρα η υλοποίησή μας δεν είναι τέλεια και υπάρχει περιθώριο βελτίωσης. Το χρονικό κομμάτι των υπολογισμών των πρακτόρων είναι ένα μέρος της εργασίας που χωράει αρκετή βελτίωση, αφού όπως είδαμε, ακόμα και με την χρήση πολυνηματικού προγραμματισμού κοστίζει αρκετά σε χρόνο. Επιπρόσθετα, ο υπολογισμός όλων των πιθανών κωδικών κοστίζει αρκετά σε μνήμη και ήταν αδύνατη η επίλυση παιχνιδιών μεγάλου μήκους. Οπότε, η επίλυση μεγαλύτερων και συνεπώς πιο δύσκολων κωδικών, είναι μία αξιόλογη μελλοντική προσθήκη. Τέλος, θα μπορούσε να προστεθεί αξιολόγηση από μία ομάδα χρηστών (user evaluation), προκειμένου να υπάρξουν βελτιώσεις στο γραφικό περιβάλλον που χρησιμοποιείται.

6.3 Τί Μάθαμε

Η προσωπική ενασχόληση με την παρούσα διπλωματική εργασία, μου φανέρωσε πόσο δύσκολο είναι για έναν άνθρωπο να προσπαθήσει να παίξει βέλτιστα οποιοδήποτε επιτραπέζιο παιχνίδι. Η μετάβαση από επίπεδο σε επίπεδο είναι φαινομενικά εύκολη και δεν διαφαίνεται κάποια ιδιαίτερη αύξηση στην δυσκολία. Στην πραγματικότητα όμως, όταν ανεβαίνουμε επίπεδο δυσκολίας οι κωδικοί αυξάνονται εκθετικά, με αποτέλεσμα να είναι πιο δύσκολο από ότι φαντάζει. Έτσι, πλέον δεν είναι καθόλου παράλογο που μέχρι και οι Grand Masters έχουν ηττηθεί από κάποιο υπολογιστή.

7 ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] "Human intelligence | Definition, Types, Test, Theories and Facts | Britannica". https://www.britannica.com/science/human-intelligence-psychology (accessed Feb. 01, 2023).

[2] Μ. Γούδας, "Η φύση και η ανάπτυξη της νοημοσύνης". [Online]. Available: Διαφάνεια 1 (uth.gr)

[3] S. J. Russell and P. Norvig, "Τεχνητή Νοημοσύνη: Μια σύγχρονη προσέγγιση", 4η αμερικάνικη έκδοση. Αθήνα: Κλειδάριθμος, 2021.

[4] Ι. Βλαχάβας, Π. Κεφαλάς, Ν. Βασιλειάδης, Φ. Κόκκορας, και Η. Σακελλαρίου, "Τεχνητή Νοημοσύνη, Δ" έκδοση. Θεσσαλονίκη: Εκδόσεις Πανεπιστημίου Μακεδονίας, 2020.

[5] J. McCarthy, "What is Artificial Intelligence?", http://jmc.stanford.edu/articles/whatisai/whatisai.pdf

[6] A. M. Turing, "I.—Computing Machinery and Intelligence Mind", vol. LIX, no. 236, pp. 433–460, Oct. 1950, doi: 10.1093/mind/LIX.236.433.

[7] Α. Καραλέγκου "Η Τεχνητή Νοημοσύνη στην Έρευνα και Εκμετάλλευση Υδρογονανθράκων". Accessed: Feb. 02, 2023. [Online]. Available: ΚΑΡΑΛΕΓΚΟΥ_ΑΙΚΑΤΕΡΙΝΗ_ΔΙΠΛΩΜΑΤΙΚΗ_2021.pdf (ntua.gr).

[8] "Marvin Minsky | American scientist | Britannica", Jan. 20, 2023. https://www.britannica.com/biography/Marvin-Lee-Minsky (accessed Feb. 03, 2023).

[9] "What is Artificial Intelligence (AI) ? | IBM". https://www.ibm.com/topics/artificial-intelligence (accessed Feb. 02, 2023).

[10] Y. Xu *et al.*, "Artificial intelligence: A powerful paradigm for scientific research", *The Innovation*, vol. 2, no. 4, p. 100179, Nov. 2021, doi: 10.1016/j.xinn.2021.100179.

[11] "Τεχνητή Νοημοσύνη - Τι είναι και γιατί έχει σημασία". https://www.sas.com/el_gr/insights/analytics/what-is-artificial-intelligence.html (accessed Feb. 02, 2023).

[12] "Αρχαίοι Έλληνες Φιλόσοφοι". https://www.greek-language.gr/digitalResources/ancient_greek/history/filosofia/page_057.html (accessed Feb. 01, 2023).

[13] C. A. Holloway,"Decision Making Under Uncertainty: Models and Choices". Englewood Cliffs, N.J: Prentice Hall, 1979.

[14] "Νοήμονες Πράκτορες". http://repfiles.kallipos.gr/html_books/93/06a-main.html (accessed Feb. 02, 2023).

[15] Α. Γεωργούλη, "Τεχνητή νοημοσύνη". 2016. Accessed: Feb. 02, 2023. [Online]. Available: http://repository.kallipos.gr/handle/11419/3381

[16] "ENIAC - CHM Revolution". https://www.computerhistory.org/revolution/birth-of-the-computer/4/78 (accessed Feb. 03, 2023).

[17] P. Bhavsar, I. Safro, N. Bouaynaya, R. Polikar, and D. Dera, "Chapter 12 - Machine Learning in Transportation Data Analytics", in *Data Analytics for Intelligent Transportation Systems*, M. Chowdhury, A. Apon, and K. Dey, Eds. Elsevier, 2017, pp. 283–307. doi: 10.1016/B978-0-12-809715-1.00012-2.

[18] "Ferranti Mark 1 - Chessprogramming wiki". https://www.chessprogramming.org/Ferranti_Mark_1 (accessed Feb. 03, 2023).

[19] A. L. Samuel, "Some Studies in Machine Learning Using the Game of Checkers", *IBM Journal of Research and Development*, vol. 3, no. 3, pp. 210–229, Jul. 1959, doi: 10.1147/rd.33.0210.

[20] G. Tesauro, "Programming backgammon using self-teaching neural nets", *Artificial Intelligence*, vol. 134, no. 1, pp. 181–199, Jan. 2002, doi: 10.1016/S0004-3702(01)00110-2.

[21] G. Tesauro, "Neurogammon Wins Computer Olympiad", *Neural Computation*, vol. 1, no. 3, pp. 321–323, Sep. 1989, doi: 10.1162/neco.1989.1.3.321.

[22] J. Schaeffer *et al.*, "Solving Checkers" *Department of Computing Science, University of Alberta Edmonton*, Alberta, Canada https://webdocs.cs.ualberta.ca/~mmueller/ps/ijcai05checkers.pdf

[23] M. Campbell, A. J. Hoane, and F. Hsu, "Deep Blue", *Artificial Intelligence*, vol. 134, no. 1, pp. 57–83, Jan. 2002, doi: 10.1016/S0004-3702(01)00129-1.

[24] "What is Watson? IBM Takes on Jeopardy", Apr. 07, 2020. https://www.ibm.com/support/pages/what-watson-ibm-takes-jeopardy (accessed Feb. 04, 2023).

[25] Ν. Περδικάρης, "Ένα πρόγραμμα Τεχνητής Νοημοσύνης νικά τους ανθρώπους στο Πόκερ", *ertnews.gr*, Jan. 31, 2017. https://www.ertnews.gr/eidiseis/epistimi/ena-programma-technitis-noimosynis-nika-tous-anthropous-sto-poker/ (accessed Feb. 04, 2023).

[26] R. Millkkulainen, "Creating Intelligent Agents in Games", in *Frontiers of Engineering: Reports on Leading-Edge Engineering from the 2006 Symposium" at NAP.edu*, 2007. doi: 10.17226/11827.

[27] J. Mycielski, "Games with Perfect Information", in *Handbook of Game Theory*, vol. 1, R. Aumann and S. Hart, Eds. Elsevier Science Publishers, 1992. [Online]. Available: https://www2.math.upenn.edu/~ted/210F10/R

[28] Κ. Παλαιοδήμος, "Διαπραγματευτικό μοντέλο πολίτη με εφορία με χρήση ευφυών πρακτόρων", Μεταπτυχιακή Διατριβή, ΑΠΘ, Θεσσαλονίκη, 2015. http://ikee.lib.auth.gr/record/270188/?ln=el

[29] J. Levin, "Games of Incomplete Information". 2022. https://web.stanford.edu/~jdlevin/Econ%20203/Bayesian.pdf

[30] "Zero-Sum Games". https://cs.stanford.edu/people/eroberts/courses/soco/projects/1998-99/game-theory/zero.html (accessed Feb. 08, 2023).

[31] B. Kooi, "Yet Another Mastermind Strategy", *ICGA Journal*, vol. 28, no. 1, pp. 13–20, Jan. 2005, doi: 10.3233/ICG-2005-28105.

[32] A. Heeffer and H. Heeffer, "Near-Optimal Strategies for the Game of Logik", *ICGA Journal*, 2007. http://logica.ugent.be/albrecht/thesis/Logik.pdf

[33] "Dowell - Defeating_Mastermind.pdf". Accessed: Feb. 08, 2023. [Online]. Available: http://mercury.webster.edu/Aleshunas/Support%20Materials/Analysis/Dowell%20-%20Defeating_Mastermind.pdf

[34] 'Dowelll - Mastermind v2-0.pdf'. Accessed: Feb. 08, 2023. [Online]. Available: http://mercury.webster.edu/aleshunas/Support%20Materials/Analysis/Dowelll%20-%20Mastermind%20v2-0.pdf

[35] J. Vomlel, "Bayesian networks in Mastermind", [Online]. Available: http://staff.utia.cas.cz/vomlel/mastermind.pdf

[36] D. E. Knuth, "The computer as Mastermind", *Journal of Recreational Mathematics*, vol. 9, no. 1, pp. 1–6, 1976. https://www.cs.uni.edu/~wallingf/teaching/cs3530/resources/knuth-mastermind.pdf

[37] R. W. Irving, "Towards an optimum Mastermind strategy", *Journal of Recreational Mathematics*, vol. 11, no. 2, pp. 81–87, 1978.

[38] A. Bestavros and A. Belal, "MasterMind: A Game of Diagnostic Strategies". Alexandria University, 1986. [Online]. Available: https://www.cs.bu.edu/fac/best/res/papers/alybull86.pdf