ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
& ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Προγραμματισμός Αυτοματοποιημένου Επιλυτή για το παιχνίδι του Ναρκαλιευτή

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Δημήτριος Ράρρας

Επιβλέπων Καθηγητής

Δρ. Μηνάς Δασυγένης

Επίκουρος Καθηγητής

Εργαστήριο Ρομποτικής, Ενσωματωμένων και Ολοκληρωμένων Συστημάτων

ΚΟΖΑΝΗ ΙΟΥΛΙΟΣ 2022

ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
& ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Programming of an automated solver for the minesweeper game

DIPLOMA THESIS

Dimitrios Rarras

Supervising Professor

Dr. Minas Dasygenis

Assistant Professor

Robotics, Embedded and Integrated Systems Laboratory

KOZANI JULY 2022

ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
& ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΗΛΩΣΗ ΜΗ ΛΟΓΟΚΛΟΠΗΣ ΚΑΙ ΑΝΑΛΗΨΗΣ ΠΡΟΣΩΠΙΚΗΣ ΕΥΘΥΝΗΣ

Δηλώνω ρητά ότι, σύμφωνα με το άρθρο 8 του Ν. 1599/1986 και τα άρθρα 2,4,6 παρ. 3 του Ν. 1256/1982, η παρούσα Διπλωματική Εργασία με τίτλο "Προγραμματισμός Αυτοματοποιημένου Επιλυτή για το παιχνίδι του Ναρκαλιευτή" καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στα πλαίσια αυτής της εργασίας και αναφέρονται ρητώς μέσα στο κείμενο που συνοδεύουν, και η οποία έχει εκπονηθεί στο Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Πανεπιστημίου Δυτικής Μακεδονίας, υπό την επίβλεψη του μέλους του Τμήματος Δρ. Μηνά Δασυγένη αποτελεί αποκλειστικά προϊόν προσωπικής εργασίας και δεν προσβάλλει κάθε μορφής πνευματικά δικαιώματα τρίτων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή / και πηγές άλλων συγγραφέων, αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή. Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τον συγγραφέα. Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και μόνο.

Υπογραφή Φοιτητή: Δημήτριος Ράρρας

Περίληψη

Με την έλευση των λύσεων που παρείχαν οι υπολογιστές στην επιστήμη, η αυτοματοποίηση θεμελίωσε τη θέση της ως βάση της αποδοτικότητας. Διαμέσου αυτοματοποιημένων επιλυτών, αντιμετωπίζονται δυναμικά προβλήματα αυξημένης πολυπλοκότητας και μεταβαλλόμενης φύσης. Παρόλο που οι επιλυτές παρέχουν λύσεις σε τέτοια προβλήματα, η μελέτη και βελτιστοποίηση τους φέρνει στην επιφάνεια νέα ερωτήματα. Μία κατηγορία των προκλήσεων είναι και η επίλυση παιχνιδιών καθώς η εκτενής μελέτη τους οδηγεί σε σύνθετα υπολογιστικά προβλήματα. Αξιοποιώντας ένα μέσο όπως τα παιχνίδια, διευκολύνεται η παρατήρηση των αποτελεσμάτων των επιλυτών. Ως αποτέλεσμα, έχουν σημειωθεί πολλά ορόσημα στην ιστορία της επίλυσης των παιχνιδιών.

Κατά την παρούσα διπλωματική εξετάζεται το παιχνίδι του Ναρκαλιευτή και υλοποιείται μια μοναδική προσέγγιση με στόχο την επέκταση προϋπαρχόντων μεθόδων οπτικής αναγνώρισης και επίλυσης. Ο Αυτοματοποιημένος επιλυτής συνδυάζει τεχνικές μηχανικής όρασης για την εύρεση του ταμπλό και την αναγνώριση της κατάστασης των επιμέρους μπλοκ. Παράλληλα, αξιοποιεί ντετερμινιστικούς αλγορίθμους για τις απλές καταστάσεις του ταμπλό ενώ μοντελοποιεί τις πιο σύνθετες ως Προβλήματα Ικανοποίησης Περιορισμών (CSP). Το μοντέλο αυτό εισάγεται στον επιλυτή CP-SAT που βασίζεται στην υβριδική τεχνική LCG, ώστε να κωδικοποιηθεί και να επιλυθεί από τον εσωτερικό επιλυτή SAT με τα ανάλογα υπολογιστικά πλεονεκτήματα. Με βάση τις λύσεις που παρέχει ο CP-SAT για τις πιθανές αναθέσεις ναρκών, γίνεται πιθανοτική ανάλυση του ταμπλό προσδίδοντας ακρίβεια στις επόμενες κινήσεις.

Η υλοποίηση της διπλωματικής παρουσιάζει τόσο τα πλεονεκτήματα της οπτικής αναγνώρισης και της επίλυσης ως CSP, όσο και τους περιορισμούς και τις προκλήσεις που ενέχουν. Ταυτόχρονα, δια μέσω πειραματικής διαδικασίας, αναλύονται οι συσχετίσεις μεταξύ μετρικών και η επιρροή της πυκνότητας ναρκών.

Λέξεις Κλειδιά: Μηχανική όραση, ψηφιακή επεξεργασία εικόνας, ανάλυση εικόνας, προβλήματα ικανοποίησης περιορισμών, C#

Abstract

With the advent of solutions provided to science by computers, automation solidified its place as the basis of performance. Through automated solvers, dynamic problems of increased complexity and changing nature are solved. Although solvers provide solutions in such problems, studying and optimizing them brings new questions to the surface. One of the challenges is solving games as their extensive studies leads to complex computing problems. Utilizing a medium such as games facilitates the observation of solver results. As a result, there have been many milestones in the history of game solving.

In the present diploma thesis, the game of Minesweeper is examined and a unique approach is implemented with the aim of extending pre-existing methods of visual parsing and solving. The Automated Solver combines machine vision techniques to find the board and identify the state of individual blocks. At the same time, it utilized deterministic algorithms for simple board states while modeling the more complex ones as Constraint Satisfaction Problems (CSP). This model is input in a CP-SAT solver based on the hybrid LCG technique, to be encoded and solved by the internal SAT solver with the corresponding computational advantages. Based on the solutions provided by CP-SAT for possible mine assignments, a possibility analysis is performed on the board giving precision to the next moves.

The implementation of the diploma thesis presents both the advantages of visual parsing and solution as a CSP, as well as the limitations and challenges involved. At the same time, through an experimental process, the correlations between metrics and the influence of mine density are analyzed.

Keywords: Machine vision, digital image processing, image parsing, constraint satisfaction problems, C#

Ευχαριστίες

Θα ήθελα να ευχαριστήσω ιδιαίτερα τον Δρ. Μηνά Δασυγένη και τον κ. Νικόλαο Μπαρά που βρέθηκαν δίπλα μου σε κάθε στάδιο της εκπόνησης της παρούσας διπλωματικής παρέχοντας καθοδήγηση. Η επιτυχής ολοκλήρωσή της δε θα ήταν δυνατή χωρίς την εποικοδομητική κριτική και τη συμβολή τους.

Θα ήθελα να επεκτείνω τις ευχαριστίες μου στην οικογένεια και στους φίλους μου που με στήριξαν κατά τη διάρκεια της εκπόνησης και παρείχαν τα μέσα για να ελέγξω την ορθότητα των μεθόδων που υλοποίησα.

Περιεχόμενα

Κατάλογος Εικόνων

Κατάλογος Πινάκων

Κατάλογος Εξισώσεων

Κατάλογος Αλγορίθμων

Κεφάλαιο 1

Εισαγωγή

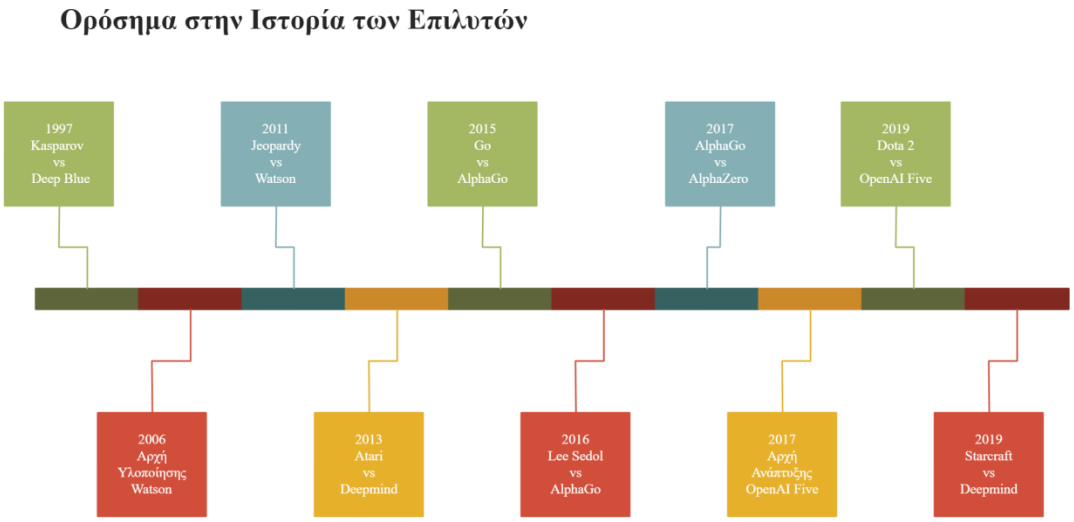
1.1 Ορισμός Προβλήματος

Η εδραίωση των υπολογιστών ως πυρήνας της τεχνολογικής ανάπτυξης αποτελεί πλέον αδιαμφισβήτητο γεγονός. Η καθημερινότητα του μέσου ανθρώπου απαρτίζεται από εργαλεία και τις λύσεις που παρέχουν, ενώ δε θα ήταν υπερβολή να γίνει η δήλωση πως οι υπολογιστές έχουν θέσει ισχυρές ρίζες σχεδόν σε κάθε πτυχή της ζωής μας. Σύμφωνα με πηγές [1], το 2021 ο μέσος άνθρωπος περνάει 6 ώρες και 57 λεπτά κάθε μέρα χρησιμοποιώντας τον υπολογιστή ή το κινητό του. Το μεγαλύτερο μέρος των ωρών αποδίδονται στην ψυχαγωγία και συγκεκριμένα, στα παιχνίδια για αυτές τις συσκευές.

Η πρόκληση των παιχνιδιών κέντριζε ανέκαθεν το ενδιαφέρον τόσο των παικτών όσο και πολλών επιστημονικών κοινοτήτων [2]. Κατά την πρώτη επαφή με το παιχνίδι, ο κάθε παίκτης ανακαλύπτει έναν ολοκαίνουργιο κόσμο με ζωηρά γραφικά, πιθανές δράσεις αλλά και κανόνες που δεν έχει ακόμα εξερευνήσει. Η αρχική γοητεία της ανακάλυψης παρέρχεται και την αντικαθιστά η έμφυτη ανάγκη του ανθρώπου να κατακτήσει την πρόκληση. Με βάση τις εμπειρίες που αποκτά σταδιακά, αλλά και με την ενδότερη κατανόηση, ο παίκτης βελτιστοποιεί τις στρατηγικές του, ακόμα κι αν δε γίνεται συνειδητά από τον ίδιο, πάντα με στόχο να καταφέρει να βγει νικητής. Υπάρχουν πολλοί παράγοντες που αλλάζουν από παιχνίδι σε παιχνίδι, όλα όμως παρέχουν μία ψυχαγωγική δοκιμασία για όσους επιδιώκουν να ακονίσουν το μυαλό τους.

Παρόλο που οι υπολογιστές έχουν προσδώσει λύσεις για αναρίθμητα προβλήματα, με την ευρύτατη μελέτη για την καινοτομία και τη βελτιστοποίηση έχουν αναδυθεί ερευνητικά ερωτήματα που απαιτούν απάντηση. Ένα από αυτά τα ερωτήματα, είναι το πως θα προγραμματιστούν υπολογιστές για να επιλύσουν ορισμένες προκλήσεις και ειδικά αυτές των παιχνιδιών. Ακόμα και όταν πρόκειται για ένα παιχνίδι με σαφείς και ξεκάθαρους κανόνες η εξέταση του συχνά οδηγεί σε πολύπλευρες και δυσεπίλυτες καταστάσεις. Τόσο από προγραμματιστικής όσο και μαθηματικής απόψεως, σχηματίζονται προβλήματα τα οποία δεν είναι εμφανή δίχως τη διερευνητική ματιά που διέπει τους δύο κλάδους. Αντιθέτως, για το μέσο παίκτη μία παρτίδα αποτελεί εναργή υπόθεση, καθώς τέτοια προβλήματα δεν έχουν κάποια επίπτωση ούτε του απαγορεύουν να τελειώσει μία παρτίδα. Η παραπλανητική αυτή

ευκολία της επίλυσης των παιχνιδιών προσελκύει ερευνητές ποικίλων επιπέδων, οι οποίοι σύντομα βρίσκονται αντιμέτωποι με γνωστά υπολογιστικά προβλήματα που καλούνται να λύσουν. Φυσικά, οι ερευνητές δεν άργησαν να ανταποκριθούν στο κάλεσμα, φτιάχνοντας πασίγνωστους επιλυτές στους οποίους κάθε σύγχρονη υλοποίηση αποδίδει ωδή [Εικόνα 1.1].



Εικόνα 1.1: Τα ορόσημα των υπολογιστών στις προκλήσεις των παιχνιδιών.

Με τη βοήθεια της τεχνητής νοημοσύνης, αναπτύχθηκαν πολλά επιφανή προγράμματα που αποδέχτηκαν την πρόκληση των παιχνιδιών ακόμη και εναντίον αληθινών παικτών, δείχνοντας σταδιακά την υπεροχή τους αντιμετωπίζοντας τους κορυφαίους παίκτες του εκάστοτε παιχνιδιού. Ίσως το πιο γνωστό παράδειγμα είναι αυτό του Kasparov εναντίον του υπολογιστή Deep Blue της IBM [3]. Σε μία ιστορική αναμέτρηση, ο πράκτορας του Deep Blue το 1997 κέρδισε τον τότε παγκόσμιο πρωταθλητή στο σκάκι Gary Kasparov πετυχαίνοντας ένα γιγαντιαίο επίτευγμα για την τεχνητή νοημοσύνη. Αφού νίκησε τον πρωταθλητή στο σκάκι, η IBM αναζήτησε την επόμενη πρόκληση. Το 2006 ξεκίνησε την ανάπτυξη του υπερυπολογιστή Watson που θα κατακτούσε το πασίγνωστο αμερικανικό παιχνίδι ερωτήσεων Jeopardy [4]. Το 2011 κατατρόπωσε τους πρωταθλητές Brad Rutter και Ken Jennings, θέτοντας ταυτόχρονα νέο ρεκόρ με 74 συνεχείς νίκες.

Το 2013, η ομάδα του Deepmind δημιούργησε ένα σύστημα που δεν επικεντρώνεται στην εκμάθηση ενός και μόνο παιχνιδιού. Χρησιμοποιώντας την οπτική αναγνώριση του παιχνιδιού κατέκτησαν κάθε παιχνίδι που διέθετε το Atari όπως Space Invaders, Pong και πολλά άλλα [5].Μερικά χρόνια μετά και συγκεκριμένα το 2015, η ομάδα του Deepmind έφερε το AlphaGo εναντίον τριών ευρωπαίων πρωταθλητών νικώντας 5-0 στο κινέζικο

παιχνίδι Go. Ένα χρόνο μετά, νίκησε τον νοτιοκορεάτη παγκόσμιο πρωταθλητή Lee Sedol, ενώ το 2017 επέκτεινε τις δυνατότητες του με τη μορφή του AlphaZero που κέρδισε τον προκάτοχο του 100-0 [6]. Την ίδια χρονιά ξεκινάει η ανάπτυξη του OpenAI Five για το πασίγνωστο παιχνίδι Dota 2. Το 2019 ήρθε στο προσκήνιο αγγίζοντας ποσοστό νίκης 99.4% καταφέρνοντας παράλληλα να κερδίσει την τότε κορυφαία ομάδα [7]. Το 2019 σήμανε μία ακόμα σημαντική χρονιά για το Deepmind με την μετάβαση του AlphaZero σε AlphaStar. Το σύστημα αυτό πέτυχε μία κατάταξη μεταξύ των 0.2% κορυφαίων παικτών του StarCraft [8].

Στην παρούσα εργασία θα αναλυθεί μία τέτοια πρόκληση με τη μορφή ενός Αυτοματοποιημένου Επιλυτή για το παιχνίδι του Ναρκαλιευτή. Ο Ναρκαλιευτής είναι ένα πασίγνωστο και κλασσικό παιχνίδι για πολλά λειτουργικά συστήματα όπως τα Windows και τα Linux. Το κλειδί για την εκτεταμένη επιτυχία του, πέρα από την ευρεία διαθεσιμότητα του, είναι το απλό σύνολο κανόνων του που επιτρέπουν στον παίκτη να αποδεχτεί άμεσα την πρόκληση. Φυσικά, όπως αναφέρθηκε και προηγουμένως, η ευκολία του παιχνιδιού είναι παραπλανητική καθώς για την αποδοτική επίλυσή του πρέπει να εφαρμοστούν και να συνδυαστούν διαφορετικές προγραμματιστικές στρατηγικές. Παρόλο που η κατανόηση και επίλυση του προβλήματος μπορούν να επιτευχθούν με βασικές γνώσεις και εκτενή μελέτη του παιχνιδιού, παραμένει ως πρόβλημα η αναγνώριση της κατάσταση του ταμπλό από τον Αυτοματοποιημένο Επιλυτή.

Σε ιδανικές συνθήκες, ένας επιλυτής έχει απόλυτο έλεγχο στο επίπεδο πρόσβασης δεδομένων (backend) του λογισμικού. Εκμεταλλευόμενος το πώς διαμορφώνεται ο κώδικας του παιχνιδιού, μπορεί να λάβει την είσοδο που χρειάζεται σε κάθε βήμα της εκτέλεσης. Σε λογισμικά που δεν διατίθεται ο πηγαίος κώδικας (closed-source) δεν είναι εφικτό για τον επιλυτή να έχει τέτοια πρόσβαση. Ως εκ τούτου, κατά την εργασία η συλλογή δεδομένων δε θα γίνεται διαμέσου κάποιας Διεπαφής Προγραμματισμού Εφαρμογών (API) ή με τη χρήση του πηγαίου κώδικα. Αντίθετα, το λογισμικό θα βασίζεται στην μηχανική όραση για την εύρεση του ταμπλό και την ερμηνεία των περιεχομένων του όπως θα έκανε ένας άνθρωπος. Αξίζει να σημειωθεί πως η υλοποίηση με μηχανική όραση έχει τόσο προτερήματα, όσο και περιορισμούς που ενέχει η μέθοδος αυτή.

1.2 Κίνητρα και Στόχοι

Η ολοένα αυξανόμενη πολυπλοκότητα των προβλημάτων που συναντώνται σε διάφορους τομείς της επιστήμης καλεί για την αυτοματοποίηση λειτουργιών. Για να περατωθούν αποδοτικά και δίχως ανθρώπινη παρέμβαση, διαμέσου επιλυτών μπορούμε να αντιμετωπίσουμε δυναμικά τέτοιες προκλήσεις. Καθώς τα προβλήματα που προαναφέρθηκαν δεν έχουν πάντα στατικές συνθήκες και μεταβλητές, πρέπει τα εργαλεία που αξιοποιούνται να προσφέρουν αυτή την ευελιξία και συνήθως αυτό επιτυγχάνεται με οπτική αναγνώριση και ανάλυση εικόνων μέσω μηχανικής μάθησης. Υπό το φως των δυνατοτήτων που μπορούν να παρέχουν, εμφανίζεται αξιοσημείωτη ζήτηση για τέτοιους αλγορίθμους τόσο στην καθημερινότητα, όσο και σε μεγαλύτερη κλίμακα [9]. Παράλληλα με την υπάρχουσα ζήτηση, υπάρχει η ανάγκη για εφαρμογές που αναδεικνύουν τις δυνατότητες των αυτοματοποιημένων επιλυτών προσελκύοντας νέους προγραμματιστές στον κλάδο έχοντας ως επακόλουθο την περαιτέρω ανάπτυξη. Επομένως, κύριο κίνητρο αποτελεί η έλλειψη μιας εφαρμογής η οποία ναι μεν υπογραμμίζει αυτές τις δυνατότητες, αλλά δίνει και μία ματιά εκ των έσω στον προγραμματισμό της για όσους επιθυμούν να το επιδιώξουν.

Κατά τη διπλωματική εργασία προγραμματίστηκε ένας επιλυτής με στόχο να λύνει παρτίδες Ναρκαλιευτή με ακρίβεια συλλέγοντας δεδομένα για το παιχνίδι αποκλειστικά με τη χρήση μηχανικής όρασης. Καίριο ρόλο είχε τόσο η ταχύτητα, όσο και η ακρίβεια του αλγορίθμου οπότε κατά την υλοποίηση έγινε κατάλληλος συνδυασμός στρατηγικών και εργαλείων για την επίτευξη και των δύο εξίσου. Ως πρώτο βήμα προς το στόχο της επίλυσης, ο αλγόριθμος αναλύει ένα στιγμιότυπο της οθόνης και εξάγει πληροφορίες σχετικά με το ταμπλό. Έπειτα, οργανώνει τις πληροφορίες που σύλλεξε και κατασκευάζει μία εσωτερική αναπαράσταση του ταμπλό. Με αυτό τον τρόπο καλύπτεται μια από τις βασικότερες πτυχές του στόχου, αυτή της αναγνώρισης της κατάστασης και της διαμόρφωσης της εισόδου των μεθόδων που ακολουθούν. Εξίσου σημαντική για την εκπλήρωση του στόχου είναι η επεξεργασία των δεδομένων από τους επιμέρους αλγορίθμους. Ύστερα από το σχηματισμό του συνόλου των μπλοκ σε μορφή αναγνώσιμη από κώδικα και αναλόγως με την κατάσταση που εκφράζουν αναλαμβάνει δράση ο ανάλογος επιλυτής. Ο αλγόριθμος αποφασίζει πάντα την επόμενη κίνηση ανεξάρτητα από το αν υπάρχουν σίγουρες ενέργειες, ενώ όπου χρειαστεί παίρνει υπολογισμένα ρίσκα. Ταυτόχρονα είναι σε θέση να αναγνωρίσει τη νίκη ή την ήττα σε κάθε μία από τις κινήσεις ώστε να τερματίσει την παρτίδα. Προτού ολοκληρωθεί η εκτέλεση και εκπληρωθεί ο στόχος του επιλυτή, αποθηκεύονται ορισμένα μετρικά για τη

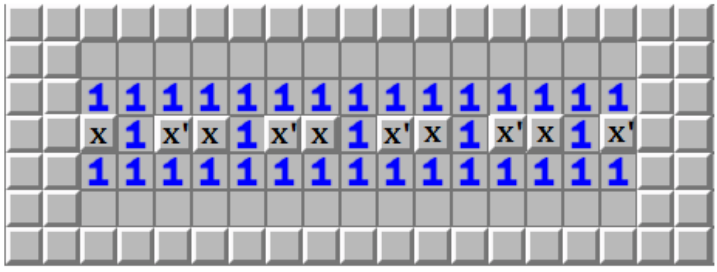
διεκπεραίωση πειραματικής διαδικασίας τα αποτελέσματα της οποίας θα εξεταστούν στο Κεφάλαιο 4.

1.3 Σχετικές Εργασίες

Ο Ναρκαλιευτής έχει τις ρίζες του στη δεκαετία του εξήντα, αλλά κυκλοφόρησε για υπολογιστή το 1989. Δεν άργησε να εγείρει το ενδιαφέρον επιστημονικών κοινοτήτων επιφέροντας επιφανείς μελέτες και αλγοριθμικές προσεγγίσεις για την επίλυσή του με τις πρώτες δημοσιεύσεις να εμφανίζονται τη δεκαετία του ενενήντα. Στις δημοσιεύσεις αυτές θεμελιώθηκαν αποδείξεις για το παιχνίδι, ενώ ανά τα χρόνια σημειώθηκε σημαντική πρόοδος όσον αφορά τις στρατηγικές επίλυσης.

1.3.1 Θεωρητική Μελέτη

Ο Richard Kaye έγινε πρωτοπόρος το 2000 φέρνοντας στο προσκήνιο το λεγόμενο πρόβλημα Συνοχής του Ναρκαλιευτή (ή απλά Συνοχής) μαζί με αποδείξεις πως ανήκει στην κλάση προβλημάτων πολυπλοκότητας NP-complete [10]. Κατά το πρόβλημα Συνοχής, πρέπει να βρεθεί αν, για ένα μερικώς αποκαλυμμένο ταμπλό, υπάρχει τρόπος τοποθέτησης των εναπομεινάντων ναρκών που να έχει συνοχή με την κατάσταση του ταμπλό. Σε αυτό το σημείο, επειδή η επιβεβαίωση του τρόπου τοποθέτησης απαιτεί πολυωνυμικό χρόνο, έχει ήδη αποδείξει ότι ανήκει στην κλάση NP. Στη συνέχεια, έδειξε πως διάφορες διαμορφώσεις του ταμπλό μιμούνται τη συμπεριφορά καλωδίου ή διανεμητή (splitter) καθώς και ορισμένες λογικές πύλες [Εικόνα 1.2]. Δείχνοντας πως αυτές οι καταστάσεις μπορούν να συνδυαστούν με οποιοδήποτε τρόπο για να φτιάξουν κάθε λογικό κύκλωμα τεκμηρίωσε πως μπορεί να γίνει αναγωγή του προβλήματος σε κύκλωμα SAT κάτι το οποίο επιλύεται σε πολυωνυμικό χρόνο άρα είναι και NP-complete.



Εικόνα 1.2 Αναπαράσταση Καλωδίου κατά τα πρότυπα του Kaye.

Τα αποτελέσματα του Kaye έκαναν ιδιαίτερα γνωστό ένα από τα μεγαλύτερα ανοικτά προβλήματα στα σύγχρονα μαθηματικά, το P vs NP [11]. Το πρόβλημα αυτό ανήκει στα επτά προβλήματα του βραβείου Μιλλένιουμ από το Ινστιτούτο Μαθηματικών Κλέι με αμοιβή ένα εκατομμύριο δολάρια για την πρώτη λύση. Το 2002 ο ίδιος απέδειξε ότι ο Ναρκαλιευτής είναι Turing-complete το οποίο σημαίνει πως στα πλαίσια του παιχνιδιού μπορεί να γραφτεί ένα πρόγραμμα που πάντα θα δώσει λύση ωστόσο δίχως εγγυήσεις για το χρόνο ή τη μνήμη που θα απαιτηθεί [12].

Το 2004, ο Pedersen συνεισέφερε στη μελέτη του Ναρκαλιευτή κάνοντας αποδείξεις στα πλαίσια των κλάσεων προβλημάτων πολυπλοκότητας DP και #P [13]. Εξηγεί ότι το να διαπιστώσουμε αν ένα ταμπλό έχει μοναδική λύση, όπως και το να ελέγξουμε αν μία κίνηση είναι ασφαλής είναι DP-complete προβλήματα χωρίς ανάγκη για ένα ταμπλό που τηρεί τη συνοχή στα πρότυπα του Kaye. Ταυτόχρονα, εισάγει το Πρόβλημα Καταμέτρησης κατά το οποίο πρέπει να υπολογιστεί ο αριθμός πιθανών τοποθετήσεων ναρκών που τηρούν τη Συνοχή κατά Kaye και αποδεικνύει ότι είναι #P-complete.

Το 2012, ο de Bondt έδωσε μια νέα ματιά στο πρόβλημα Συνοχής του Ναρκαλιευτή τεκμηριώνοντας ότι είναι κι αυτό NP-complete για τετράγωνα ταμπλό (n×n) με την προϋπόθεση πως η αρχική κίνηση αποκαλύπτει μόνο ένα μπλοκ [14]. Επιπρόσθετα, έκανε ανάλυση της πολυπλοκότητας της διαδικασίας του παιχνιδιού δεδομένου ότι σε κάθε κίνηση οι εναπομένουσες διαμορφώσεις των ναρκών που τηρούν τη Συνοχή κατά Kaye είναι ισοπίθανες.

1.3.2 Αλγόριθμοι και Επιλυτές

Η μελέτη των πιθανών στρατηγικών, αλλά και η προγραμματιστική πρόκληση που παρουσιάστηκε με την κυκλοφορία του Ναρκαλιευτή είχε ως αποτέλεσμα διαφορετικές προσεγγίσεις ανά τα χρόνια. Οι προσεγγίσεις είτε είχαν ως σκοπό το μεγαλύτερο ποσοστό επιτυχίας, είτε να προσδώσουν μία ολοκαίνουργια ματιά συμπληρώνοντας προϋπάρχοντα έργα. Οι επικρατέστεροι αλγόριθμοι θα εξεταστούν με μεγαλύτερη λεπτομέρεια στο Κεφάλαιο 3.

Μία από τις πρώτες υλοποιήσεις έκανε ο Adamatzky [15] το 1997 με τη μορφή ενός διακριτού υπολογιστικού μοντέλου που ονομάζεται Κυτταρικό Αυτόματο. Καινοτόμες προσεγγίσεις είχαν ο Kamenetsky [16] και ο Golan [17] με γραφικά μοντέλα, ο Nakov με

ενισχυτική μάθηση [18] και οι Peña και Wrobel με αλγόριθμους μάθησης [19]. Εξίσου μοναδικές ήταν οι μέθοδοι των Vomlelova και Vomlel με Bayesian δίκτυα [20] καθώς και του Buffet με μη-ντετερμινιστικά δέντρα άνω ορίου εμπιστοσύνης (UCT) [21]. Όλες αυτές οι υλοποιήσεις απέκλιναν από την πλειοψηφία που κυρίως επέλεξε να επιλύσει το Ναρκαλιευτή με προγραμματισμό περιορισμών (Constraint Programming) και με ντετερμινιστικές ευριστικές μεθόδους.

Όσον αφορά την επίλυση με βάση τους περιορισμούς, οι Bayer [22], Collet [23] και Pedersen [13] προγραμμάτισαν απλούστερους επιλυτές με μεγαλύτερη ταχύτητα λαμβάνοντας υπόψη μόνο τοπικούς περιορισμούς. Στο κομμάτι που τους ξεπερνούσαν με ευκολία πιο σύνθετες υλοποιήσεις ήταν το ποσοστό επιτυχίας. Οι επιλυτές που λάμβαναν υπόψη όλους τους περιορισμούς του ταμπλό και έλυναν για όλες τις πιθανές λύσεις όπως αυτοί του Studholme [24] και του Legendre [25] γρήγορα έγιναν οι πιο δημοφιλείς.

Εκτεταμένη χρήση βρήκαν και οι ευριστικές μέθοδοι, με τον Studholme [24] να τις μελετά και να αναπτύσσει μία που προτιμά την αποκάλυψη μπλοκ που βρίσκονται στις γωνίες και στις άκρες του ταμπλό. Ο Legendre [25] συνδύασε αλγορίθμους που μεγιστοποιούν ή ελαχιστοποιούν την απόσταση από τις γωνίες ή τις άκρες του ταμπλό κάνοντας δοκιμές για την απόδοσή τους. Αξιοσημείωτη πρόοδο σημείωσε η Jinzheng Tu [26] αναλύοντας τη θεωρητικά βέλτιστη στρατηγική που πραγματεύεται ο Golan [27] με συνδυασμό ευριστικών μεθόδων και εκτενών προσομοιώσεων για να δημιουργήσει, σε δικούς της όρους, τη σχεδόν βέλτιστη στρατηγική.

Ιδιαίτερη σημασία για κάθε επιλυτή που επιδιώκει μεγάλο ποσοστό επιτυχίας έχει η αρχική κίνηση. Ο Studholme [24], ο Beccera [28] και η Tu [26] υποστηρίζουν στις εργασίες τους πως η αρχή από τις γωνίες είναι η βέλτιστη ενώ αναλύουν ενδελεχώς τον αντίκτυπο τόσο στην ασφάλεια της δεύτερη κίνησης όσο και στην έκβαση του παιχνιδιού.

Κοινός παράγοντας των περισσότερων επιλυτών είναι το ότι επικεντρώνονται στην επεξεργασία των δεδομένων του ταμπλό και όχι στη συλλογή τους. Ο Packard [29] έφτιαξε έναν αλγόριθμο με έμφαση στην επίλυση του Ναρκαλιευτή χωρίς να είναι δεδομένη η πρόσβαση στον κώδικα του παιχνιδιού. Αξιοποιώντας εικόνες επιμέρους καταστάσεων των μπλοκ καταφέρνει την οπτική ανάλυση του ταμπλό με μεγάλη ακρίβεια.

Επεκτείνοντας τη μέθοδο του Packard, η παρούσα εργασία θα επιλύσει το Ναρκαλιευτή με μεθόδους μηχανικής όρασης για τη συλλογή πληροφοριών και την

αναγνώριση της κατάστασης του παιχνιδιού με μεγαλύτερη απόδοση. Ταυτόχρονα, θα αξιοποιηθούν ευριστικές μέθοδοι και τεχνικές προγραμματισμού περιορισμών για την επίτευξη ικανοποιητικού ποσοστού επιτυχίας. Ο συνδυασμός αξιόπιστης εισόδου και ταχείας επίλυσης θα προσδώσουν έναν ευέλικτο επιλυτή για το παιχνίδι του Ναρκαλιευτή.

1.4 Οργάνωση Κεφαλαίων

Στο παρόν κεφάλαιο διατυπώνεται το πρόβλημα που καλείται να εξετάσει η διπλωματική εργασία, εκφράζονται τα κίνητρα και οι στόχοι της ενώ γίνονται και αναφορές σε παρόμοια έργα με τις ανάλογες υλοποιήσεις τους. Στο Κεφάλαιο 2, παρουσιάζεται το θεωρητικό υπόβαθρο της εργασίας και αναλύονται οι αλγόριθμοι που τίθενται σε χρήση για την αναγνώριση και την επίλυση του Ναρκαλιευτή. Παράλληλα, αναφέρεται η πλατφόρμα ανάπτυξης του λογισμικού με τα εργαλεία που χρησιμοποιήθηκαν. Στο Κεφάλαιο 3, αναπτύσσεται ο συνδυασμός από μεθοδολογίες για την αναγνώριση και επίλυση καταστάσεων του ταμπλό. Στο Κεφάλαιο 4, αναλύεται η διαδικασία της πειραματικής διαδικασίας για τη συλλογή στατιστικών στοιχείων, ενώ διατυπώνονται και οι συσχετίσεις μεταξύ ιδιοτήτων του ταμπλό. Τέλος, το Κεφάλαιο 5 είναι ο επίλογος της εργασίας όπου παρουσιάζονται τα συμπεράσματα της εργασίας, ενώ προτείνονται μελλοντικές επεκτάσεις ή βελτιώσεις του αλγορίθμου.

Κεφάλαιο 2

Θεωρητικό Υπόβαθρο

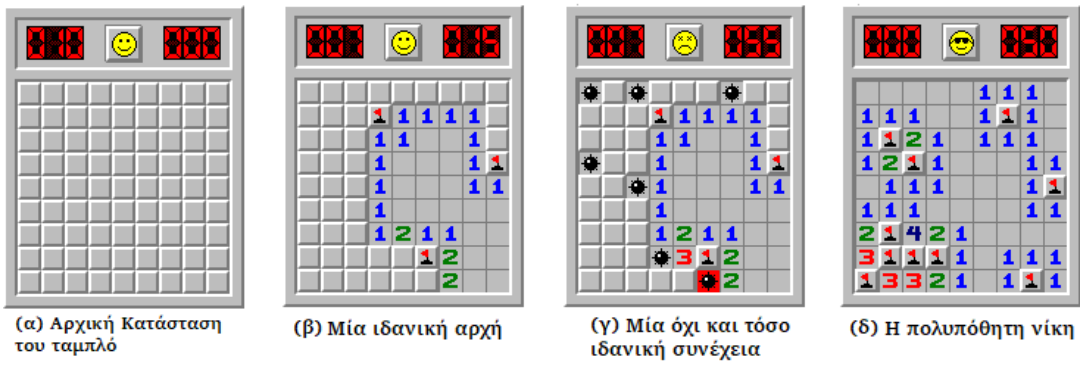
Σε αυτό το κεφάλαιο θα παρατεθούν μερικοί από τους όρους που χρησιμοποιούνται στην εργασία, καθώς και ορισμένα προβλήματα και αλγόριθμοι. Με αυτόν τον τρόπο, θα τεθούν οι βάσεις για την καλύτερη κατανόηση και την απρόσκοπτη ροή.

2.1 Αυτοματοποιημένος Επιλυτής

Στον προγραμματιστικό κόσμο, οι επιλυτές αποτελούν μία εφαρμογή η οποία ενσωματώνει έναν ή περισσότερους αλγορίθμους για να λύσουν διάφορες κλάσεις προβλημάτων. Δέχονται μία αρχική κατάσταση ή ένα σύνολο δεδομένων εισόδου που τηρούν ορισμένους κανόνες, οι οποίοι διαμορφώνονται κατά την αναγνώριση του τύπου του προβλήματος. Ως Αυτοματοποιημένος Επιλυτής για την παρούσα εργασία ορίζεται ο αλγόριθμος ο οποίος, με τον κατάλληλο προγραμματισμό, μπορεί να επιβεβαιώσει τον τύπο αυτό για να επιλέξει τη σωστή στρατηγική επίλυσης και να συγκροτήσει την απαιτούμενη είσοδο χωρίς παρέμβαση από το χρήστη.

2.2 Ναρκαλιευτής

Η παρτίδα ξεκινάει και το ταμπλό καλύπτεται από ένα πλέγμα m×n διαστάσεων με κλειστά μπλοκ [Εικόνα 2.1α]. Κάνοντας την πρώτη κίνηση, ο παίκτης αποκαλύπτει το πρώτο μπλοκ της επιλογής του και βρίσκεται αντιμέτωπος με μία κατάσταση του ταμπλό όπως στην Εικόνα 2.1β. Τα νούμερα που περιέχουν ορισμένα ανοικτά μπλοκ συμβολίζουν ότι στα γειτονικά κελιά κρύβεται αντίστοιχος αριθμός ναρκών και έχουν εύρος από ένα έως και οκτώ. Σε κάθε κίνηση, ο παίκτης πρέπει να λάβει υπόψη του κάθε αριθμό που γειτονεύει με ένα μπλοκ προτού αποφασίσει ότι το εκάστοτε μπλοκ είναι ασφαλές, ειδάλλως μπορεί να αποκαλύψει νάρκη φέρνοντας το πρώιμο τέλος του παιχνιδιού όπως στην Εικόνα 2.1γ. Καλή πρακτική είναι η σήμανση των ναρκών ώστε να αποφεύγονται λάθη, αλλά δεν απαιτείται για την ολοκλήρωση μίας παρτίδας. Ο παίκτης κερδίζει όταν όλα τα μπλοκ που δεν περιέχουν νάρκη έχουν αποκαλυφθεί, όπως στην Εικόνα 2.1δ.



Εικόνα 2.1: Βασικές Καταστάσεις του Ναρκαλιευτή (α) Η αρχική κατάσταση προτού γίνουν κινήσεις (β) Η πρώτη κίνηση με επισημάνσεις ναρκών (γ) Αποκάλυψη νάρκης (δ) Η νίκη με την αποκάλυψη όλων των κενών μπλοκ.

2.2.1 Επίπεδα Δυσκολίας

Ένας νέος παίκτης μέσα σε λίγες παρτίδες θα έχει καταλάβει πως με τη χρήση των αριθμών μπορεί να συμπεράνει ποια μπλοκ περιέχουν νάρκες και να τις επισημάνει. Σε μεγαλύτερες δυσκολίες του Ναρκαλιευτή και όσο ανεβαίνει η πυκνότητα των ναρκών αυτό δεν βρίσκει πάντα εφαρμογή. Αυτό σημαίνει πως οι νάρκες δε μπορούν να βρεθούν ντετερμινιστικά και έτσι ο παίκτης πρέπει να καταφύγει στο να μαντέψει. Υπάρχουν τρία επίπεδα δυσκολίας με κρίσιμη διαφορά την πυκνότητα των ναρκών [Πίνακας 2.1]. Στη χαμηλότερη δυσκολία, ο παίκτης σπάνια θα χρειαστεί να μαντέψει, ενώ τυπικά θα αντιμετωπίσει αριθμούς μεταξύ του 1 και του 3. Καθώς αυξάνεται η δυσκολία θα συναντήσει όλο και περισσότερους μεγάλους αριθμούς, αλλά κυρίως θα βρεθεί συχνότερα σε καταστάσεις που πρέπει να μαντέψει. Ακόμα και για τους έμπειρους παίκτες, η έλλειψη πληροφοριών για την επιλογή της επόμενης κίνησης είναι μία κατάσταση που πρέπει να αποφευχθεί καθώς αφαιρεί τον έλεγχο της παρτίδας.

Δυσκολία	**Μέγεθος Πλέγματος**	**Διαστάσεις σε pixels**	**Αριθμός Ναρκών**	**Πυκνότητα (νάρκες/μπλοκ)**
Beginner	9×9	144×144	10	≈0,1235
Intermediate	16×16	256×256	40	0,15625
Expert	30×16	480×256	99	0,20625

Πίνακας 2.1: Επίπεδα Δυσκολίας Ναρκαλιευτή.

2.2.2 Προσέγγιση των Παικτών

Από την κυκλοφορία του παιχνιδιού, ξεκίνησε σταδιακά η αναζήτηση για αποδοτικές προσεγγίσεις με μοτίβα που να ελαχιστοποιούν την ανάγκη για τυχαίες κινήσεις και κατά

συνέπεια το ρίσκο κάθε παρτίδας. Η αναζήτηση αυτών των στρατηγικών και ο ανταγωνισμός μεταξύ παικτών γεννήθηκαν από την έλευση κατατάξεων σε διάφορες ιστοσελίδες καθώς και από την αναγνώριση τους από την επιτροπή των ρεκόρ Γκίνες. Για παράδειγμα, αξιοσημείωτο επίτευγμα αναγνωρισμένο από την επιτροπή έθεσε το 2014 ο Kamil Murański [30] με συνολικό χρόνο 38.65 δευτερόλεπτα για την επίλυση και των τριών επιπέδων δυσκολίας.

Οι παίκτες κατά πλειοψηφία επιδιώκουν να εξακριβώσουν τις σίγουρες κινήσεις στο γρηγορότερο δυνατό χρόνο, ενώ οι κινήσεις που απαιτούν περαιτέρω εξέταση συνήθως περατώνονται δίχως ιδιαίτερη ανάλυση ή ακρίβεια. Αυτό συμβαίνει διότι οι κατατάξεις είναι αποκλειστικά με βάση την ταχύτητα. Η ακρίβεια έρχεται ως υποπροϊόν, αφού οι λιγότερες κινήσεις συνήθως σημαίνουν καλύτερο χρόνο. Κατά επέκταση, αυτό σημαίνει πως η προσέγγιση των παικτών δε δίνει βάση στο να λύσει κάθε παρτίδα επιτυχώς, αλλά να λύσει την παρτίδα στο γρηγορότερο χρόνο ανεξάρτητα από το πόσες προσπάθειες θα πάρει. Σε δεδομένα στρατηγικής επίλυσης, αυτό υποδηλώνει πως σε ορισμένες σύνθετες καταστάσεις (όπως σε όσες δεν υπάρχει απόλυτα σίγουρη κίνηση) ο παίκτης δε θα προσπαθήσει να μεγιστοποιήσει λογικά τις πιθανότητες του και θα διαλέξει όσο πιο γρήγορα μπορεί μία από τις διαθέσιμες κινήσεις. Επίσης για λόγους ταχύτητας, οι τακτικές No-Flag, κατά τις οποίες ο παίκτης δεν κάνει σήμανση των ναρκών, είναι επιφανείς μεταξύ έμπειρων παικτών.

2.2.3 Παίκτης εναντίον Επιλυτή

Αναμφισβήτητα τα θεμέλια της προγραμματιστικής προσέγγισης είναι η ανθρώπινη σκέψη σχετικά με την επίλυση του Ναρκαλιευτή. Η διαφορά των επιλυτών έρχεται με τη μορφή των επεκτάσεων των ανθρώπινων δυνατοτήτων και την ικανότητα να αποθηκεύει, να επεξεργάζεται και να αποτυπώνει εσωτερικά την κατάσταση του ταμπλό. Για παράδειγμα, στα σημεία που δεν υπάρχουν σίγουρες κινήσεις και ο παίκτης έχει μεγάλη πιθανότητα να αποκαλύψει νάρκη, ο επιλυτής μπορεί να βρει την πιο ασφαλή κίνηση βάσει πιθανοτήτων. Είναι ξεκάθαρο πως, με κατάλληλο προγραμματισμό, οι επιλυτές έχουν το πάνω χέρι όσον αφορά την ακρίβεια και, ως εκ τούτου, στις ελάχιστες κινήσεις κάθε παρτίδας. Εξίσου προφανές είναι πως η ταχύτητα λήψης αποφάσεων, αλλά και έκφρασης τους στο ταμπλό είναι αστραπιαία συγκριτικά με των παικτών.

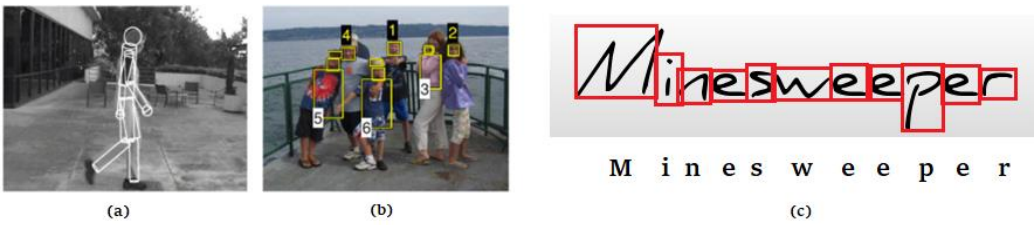
Τα γεγονότα αυτά καθιστούν προφανές ότι οι αληθινοί παίκτες δε μπορούν να συναγωνιστούν τον υπολογιστή στο Ναρκαλιευτή. Οπότε, φυσικά, δε μπορεί να τεθεί και σύγκριση μεταξύ επιλυτή και παίκτη όπως στις περιπτώσεις που αναλύθηκαν στο εισαγωγικό κεφάλαιο. Αντί αυτού, οι συγκρίσεις προσεγγίσεων γίνονται μεταξύ επιλυτών και των μεθόδων που επιλέγουν για να αναλύσουν και να αντιμετωπίσουν την πρόκληση. Πιο συγκεκριμένα, η σύγκριση γίνεται με βάση το ποσοστό επιτυχίας (win rate) ή την καινοτομία της προσέγγισης.

2.3 Μηχανική Όραση

Βρίσκοντας γόνιμο έδαφος για την ταχύτατη ανάπτυξή της, η τεχνητή νοημοσύνη έθεσε τα θεμέλια της σε μια πληθώρα επιστημονικών πεδίων. Ένα από αυτά τα πεδία είναι η μηχανική όραση κατά την οποία προσεγγίζεται η αλγοριθμική μελέτη της όρασης [31]. Με τη συλλογή, επεξεργασία και ανάλυση ψηφιακών φωτογραφιών έχει σκοπό να παράγει συμβολικά δεδομένα με βάση το περιεχόμενο ή να αλλάξει τις ιδιότητες τους. Όπως ένας άνθρωπος μπορεί να αποκτήσει ένα οπτικό ερέθισμα με την αίσθηση της όρασης, ένα πρόγραμμα μπορεί να έχει την ανάλογη είσοδο με κάποιον αισθητήρα. Έγκυρες είσοδοι για έναν αλγόριθμο μηχανικής όρασης είναι φωτογραφίες, βίντεο ακόμα και τρισδιάστατες αναπαραστάσεις. Σε κάθε περίπτωση, υπάρχουν υποσύνολα της μηχανικής όρασης που χειρίζονται τις ανάλογες εισόδους.

Μερικές από τις πιο διαδεδομένες εφαρμογές που συνδυάζουν τεχνικές μηχανικής όρασης:

- Εκτίμηση 3D μοντέλων [Εικόνα 2.2a][32]
- Αναγνώριση προσώπων [Εικόνα 2.2b][33]
- Οπτική Αναγνώριση Χαρακτήρων (OCR) [Εικόνα 2.2c] [34]
- Ανάλυση απεικονίσεων με ακτίνες X [35]
- Αυτοματοποίηση και εποπτεία ρομπότ παραγωγής [36]
- Αυτοοδηγούμενα Αυτοκίνητα και εντοπισμός πεζών [37]



(a) (b) (c)

Εικόνα 2.2: Εφαρμογές της Μηχανικής Όρασης (a) Εκτίμηση στάσης του σώματος [32] **(b) Αναγνώριση Προσώπων** [33] **(c) Οπτική Αναγνώριση Χαρακτήρων.**

2.3.1 Ψηφιακή Επεξεργασία Εικόνας

Υποσύνολο της μηχανικής όρασης που χειρίζεται δισδιάστατες απεικονίσεις είναι η Ψηφιακή Επεξεργασία Εικόνας. Η εικόνα μπορεί να ορίζεται ως μία δισδιάστατη συνάρτηση $f(x, y)$, όπου x και y είναι συντεταγμένες και το πλάτος του f σε οποιοδήποτε ζεύγος (x, y) λέγεται ένταση για το εκάστοτε σημείο της εικόνας. Όταν οι τιμές του x και y είναι διακριτές ποσότητες έχουμε μία ψηφιακή εικόνα. Όλα τα εικονοστοιχεία στην ψηφιακή εικόνα έχουν συγκεκριμένη τοποθεσία και τιμή και κάθε συνάρτηση αξιοποιεί τα δεδομένα αυτά. Η επεξεργασία μίας ψηφιακής εικόνας χωρίζεται σε τρία επίπεδα [38]:

1. Χαμηλού επιπέδου θεωρείται η προεπεξεργασία της εικόνας όπως η μείωση του θορύβου. Η είσοδος και έξοδος είναι πάντα εικόνες.
2. Η επεξεργασία μεσαίου επιπέδου περιέχει εργασίες όπως τμηματοποίηση της εικόνας και αναγνώριση αντικειμένων. Η είσοδος συνήθως είναι εικόνα αλλά οι έξοδοι συχνά είναι χαρακτηριστικά της εικόνας.
3. Υψηλού επιπέδου χαρακτηρίζεται η ομαδοποίηση αντικειμένων και η εξαγωγή δεδομένων από αυτά.

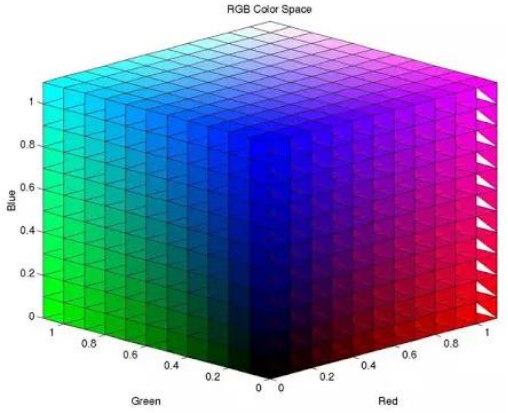
2.3.1.1 Μετατροπή χρωματικού μοντέλου

Μία από τις βασικότερες συναρτήσεις χαμηλού επιπέδου είναι αυτή της αλλαγής χρωματικού μοντέλου για μία φωτογραφία. Ως χρωματικό μοντέλο ορίζεται ένα αφαιρετικό μαθηματικό μοντέλο που περιγράφει τα χρώματα ως σύνολα τιμών με συγκεκριμένο εύρος. Για παράδειγμα, ένα από τα πιο γνωστά χρωματικά μοντέλα είναι το RGB (Red Green Blue) κατά το οποίο όλα τα χρώματα κωδικοποιούνται βάσει συνδυασμού κόκκινου, πράσινου και μπλε. Σε αυτό το μοντέλο, το χρώμα αναπαριστάται ως μία τριάδα αριθμών με εύρος από το 0 έως το 255 και δημιουργείται με την αποτύπωση της έντασης των βασικών χρωμάτων και την υπέρθεση τους [Εικόνα 2.3][39].

Για ορισμένες συναρτήσεις μεσαίου επιπέδου είναι απαραίτητη η μετατροπή του χρωματικού μοντέλου σε κλίμακες του γκρι (Grayscale). Αυτό είναι εφικτό με τη σταθμισμένη μέθοδο (γνωστή και ως μέθοδος φωτεινότητας) η οποία λαμβάνει υπόψη την ανθρώπινη αντίληψη του μήκους κύματος των χρωμάτων και ειδικά το γεγονός ότι το ανθρώπινο μάτι είναι πιο ευαίσθητο στο πράσινο χρώμα.

$$\boldsymbol{RGB[A]\ to\ Gray: Y \leftarrow 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B} \quad \textbf{(2.1)}$$

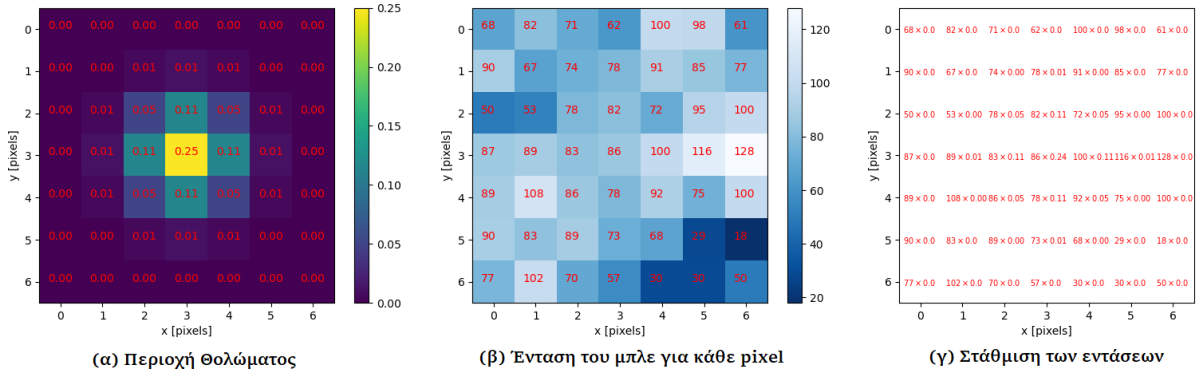
Κατά την εξίσωση 2.1, γίνεται ένας υπολογισμός για κάθε ένα από τα εικονοστοιχεία της ψηφιακής φωτογραφίας συναρτήσει της στάθμισης της έντασης του κάθε χρώματος [39].



Εικόνα 2.3: Χρωματικό Μοντέλο RGB με υπέρθεση σταθμισμένων χρωμάτων [OpenCv- Image Color Space].

2.3.1.2 Γκαουσιανό Θόλωμα

Για την αποδοτικότερη τμηματοποίηση μίας φωτογραφίας που δεν απαιτεί μεγάλη λεπτομέρεια μπορούν να εφαρμοστούν διάφορα φίλτρα χαμηλής διέλευσης (Low Pass Filters) όπως το γκαουσιανό θόλωμα [40]. Η διαδικασία του θολώματος περιλαμβάνει ένα πλέγμα εικονοστοιχείων $n \times n$ και τη μαθηματική τεχνική της συνέλιξης (convolution). Κατά την εκτέλεσή, αρχικά επιλέγεται μία περιοχή ορισμένων διαστάσεων γύρω από το εικονοστοιχείο που εξετάζεται και εξάγονται οι εντάσεις των χρωμάτων για κάθε ένα που το περιβάλλει Εικόνα 2.4β. Τα εικονοστοιχεία κοντά στο κέντρο της περιοχής έχουν μεγαλύτερο συντελεστή ως προς το τελικό αποτέλεσμα, όπως παρουσιάζεται στην Εικόνα 2.4α. Τέλος, η νέα τιμή έντασης για το εξεταζόμενο εικονοστοιχείο είναι ο μέσος όρος των σταθμισμένων υπολογισμών εντάσεως (για κάθε χρώμα του μοντέλου ξεχωριστά) όπως στην Εικόνα 2.4γ.



(α) Περιοχή Θολώματος (β) Ένταση του μπλε για κάθε pixel (γ) Στάθμιση των εντάσεων

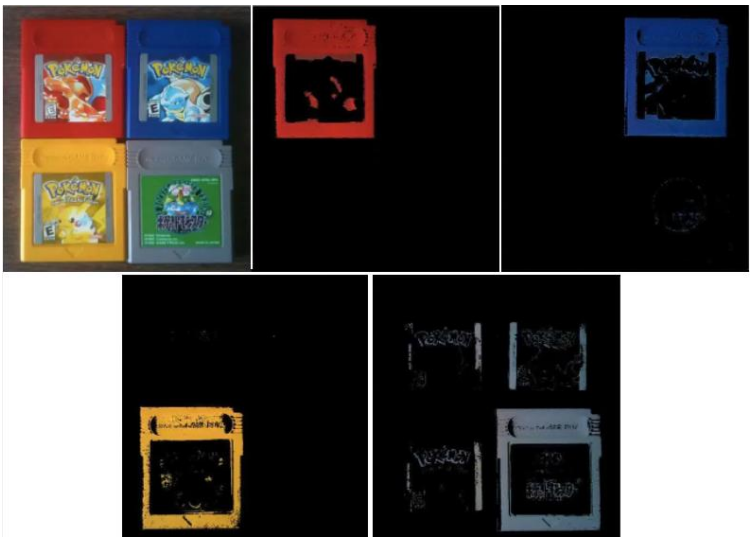
Εικόνα 2.4: Στάδια Γκαουσιανού Θολώματος [Image Processing- Blurring Images].

2.3.1.3 Κατωφλίωση

Μία κατηγορία συναρτήσεων χαμηλού επιπέδου είναι αυτή της κατωφλίωσης (thresholding) [38]. Προαπαιτούμενο για την εφαρμογή της συνάρτησης είναι η ψηφιακή εικόνα να είναι σε κλίμακα του γκρι. Σε κάθε εικονοστοιχείο της ψηφιακής εικόνας εφαρμόζεται ένα κατώφλι ή κάτω όριο. Αν η τιμή της έντασης του εικονοστοιχείου είναι κάτω από το όριο τότε του ανατίθεται 0 ή 255 αναλόγως με την τύπο κατωφλίωσης. Είναι προφανές, αφού οι πιθανές τιμές των εικονοστοιχείων της τελικής φωτογραφίας είναι δύο, πως εξάγεται μία δυαδική εικόνα.

$$\boldsymbol{Αποτέλεσμα}(x,y) = \begin{cases} \mathbf{255} & \boldsymbol{αν\ Είσοδος}(x,y) > \boldsymbol{κάτω\ όριο} \\ \mathbf{0} & \boldsymbol{αν\ Είσοδος}(x,y) < \boldsymbol{κάτω\ όριο} \end{cases} \quad \textbf{(2.2)}$$

Επέκταση των μεθόδων κατωφλίωσης είναι οι συναρτήσεις εύρους, κατά τις οποίες γίνεται δυνατή η ανίχνευση των εικονοστοιχείων μεταξύ ενός εύρους τιμών χρωματικού μοντέλου BGR (Blue Green Red). Εισάγοντας την ελάχιστη και μέγιστη επιθυμητή τιμή έντασης του κάθε χρώματος απομονώνονται οι συστοιχίες εικονοστοιχείων που είναι εντός εύρους [38][Εικόνα 2.5].

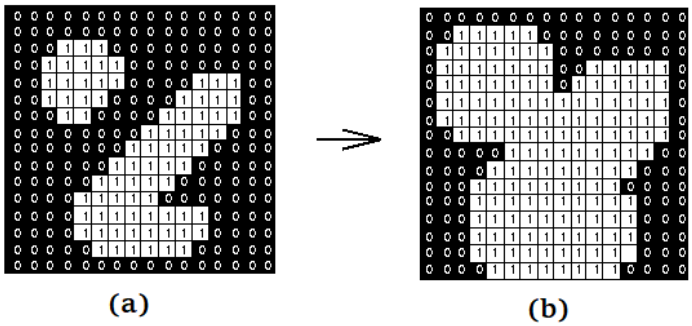


Εικόνα 2.5: Εντοπισμός Χρωμάτων με cv2.inRange [OpenCv and Python Color Detection].

2.3.1.4 Διαστολή

Στην κατηγορία των μορφολογικών λειτουργιών ανήκει η διαστολή (dilation). Κύρια χρήση είναι η αφαίρεση θορύβου από τη φωτογραφία ώστε τα αντικείμενα που θέλει ο χρήστης να απομονώσει να είναι πιο ευδιάκριτα. Κατά την εκτέλεση μίας συνάρτησης

διαστολής, γίνεται συνέλιξη της αρχικής εικόνας με μία περιοχή ορισμένου μεγέθους $n \times n$. Η περιοχή αυτή έχει ένα κεντρικό σημείο στο οποίο γίνονται οι αλλαγές. Σταδιακά, κάθε εικονοστοιχείο της εικόνας γίνεται κεντρικό σημείο της συνάρτησης, έως το τέλος της εκτέλεσης. Για κάθε περιοχή, υπολογίζεται η μέγιστη τιμή έντασης και αντικαθιστά την τιμή του εκάστοτε κεντρικού εικονοστοιχείου. Αυτό στην ουσία εκλαμβάνεται από το ανθρώπινο μάτι σαν διαστολή [Εικόνα 2.6][38].



Εικόνα 2.6: Απεικόνιση Διαστολής με τιμές έντασης (a) Αρχική Κατάσταση (b) Αποτέλεσμα [HIPR2: Dilation].

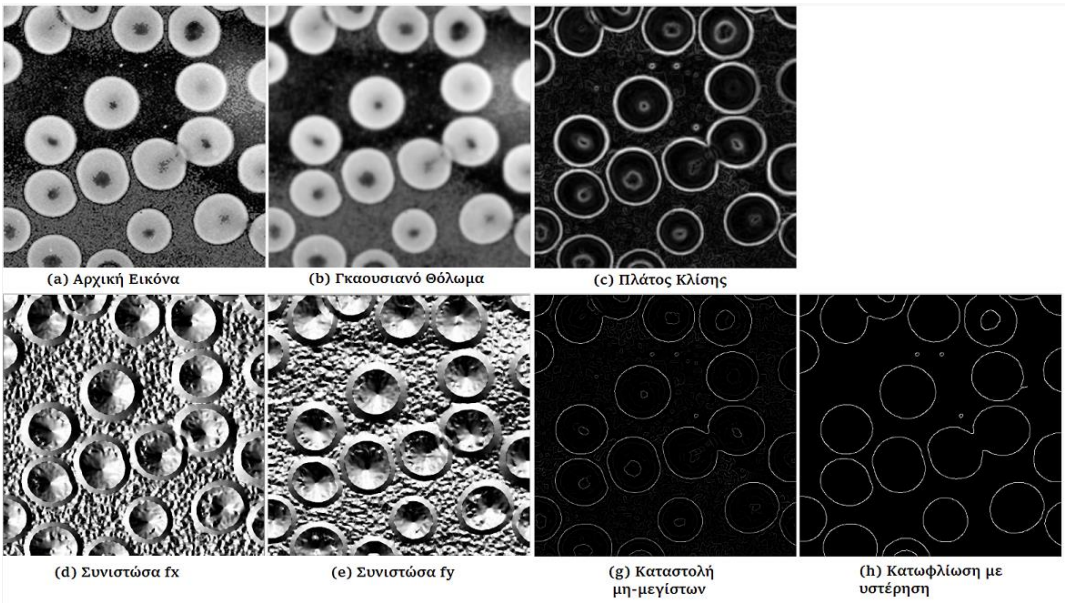
2.3.1.5 Ανίχνευση Ακμών Canny

Ο αλγόριθμος ανίχνευσης ακμών Canny αναπτύχθηκε από τον John F. Canny το 1986 με στόχο την εξαγωγή χρήσιμων δομικών πληροφοριών από μία ψηφιακή φωτογραφία [41]. Με τη διαδικασία αυτή, επίσης, μειώνεται ο όγκος των πληροφοριών που πρέπει να επεξεργαστούν σε μετέπειτα ανάλυση της φωτογραφίας. Αξίζει να σημειωθεί πως, πριν την εκτέλεση, είναι απαραίτητη η προεπεξεργασία με Γκαουσιανό Θόλωμα καθώς η διαδικασία είναι ευαίσθητη στο θόρυβο [Εικόνα 2.7a-b].

Πρώτο βήμα της ανίχνευσης είναι η εύρεση της κλίσης της έντασης για κάθε εικονοστοιχείο. Αυτό πραγματοποιείται με την εφαρμογή φίλτρου Sobel σε οριζόντια και κάθετη διεύθυνση, επιφέροντας τις δύο αντίστοιχες συνιστώσες f_x και f_y. Με αυτές τις δύο συνιστώσες μπορεί να βρεθεί το πλάτος και η γωνία της κλίσης [Εικόνα 10c-e].

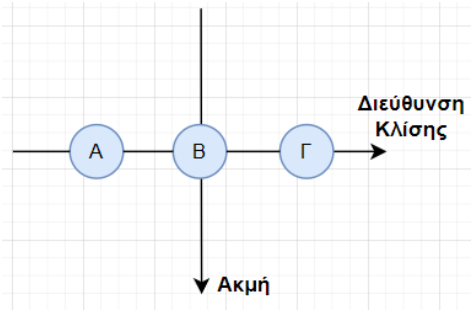
$$Πλάτος\ Κλίσης\ e = \sqrt{f_x^2 + f_y^2} \quad \textbf{(2. 3)} \qquad Γωνία\ (\theta) = tan^{-1}\left(\frac{f_y}{f_x}\right) \quad \textbf{(2. 4)}$$

Αφού γίνουν οι υπολογισμοί, ελέγχεται συνολικά η φωτογραφία για να αφαιρεθούν πιθανά εικονοστοιχεία που δεν είναι πιθανό να αποτελέσουν ακμή. Ο έλεγχος μπορεί να γίνει με την καταστολή μη-μεγίστων, καθορίζοντας αν το πλάτος του εκάστοτε εικονοστοιχείου είναι τοπικό μέγιστο κατά τη διεύθυνση της έντασης [Εικόνα 2.7g].



Εικόνα 2.7: Τα στάδια της ανίχνευσης ακμών Canny [Canny Edge Detector].

Στην Εικόνα 2.8 εξετάζεται ο μηχανισμός της καταστολής μη-μεγίστων. Το σημείο Β πάνω στη διεύθυνση της κλίσης και είναι υπό εξέταση για το αν αποτελεί ακμή. Τα γειτονικά σημεία Α και Γ βρίσκονται στην ίδια διεύθυνση κλίσης και συγκρίνονται με το Β για να κριθεί αν αποτελεί τοπικό μέγιστο. Αν αυτό ισχύει, διατηρείται για το επόμενο στάδιο ειδάλλως καταστέλλεται με την ανάθεση μηδενικής έντασης.

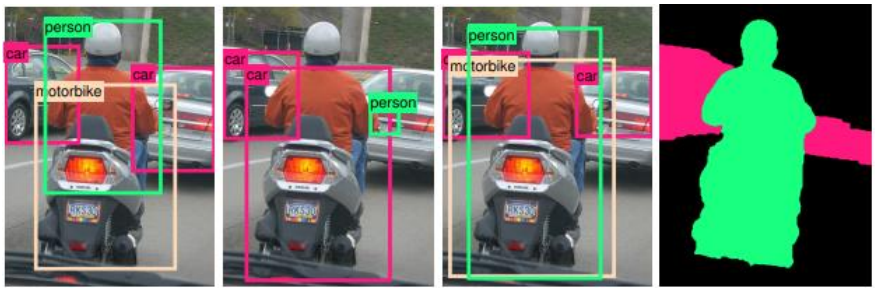


Εικόνα 2.8: Υπολογισμός μέγιστου μεταξύ γειτονικών εικονοστοιχείων για καταστολή μη-μέγιστων.

Τελευταίο στάδιο, το οποίο αποφασίζει την τελική έξοδο, είναι η κατωφλίωση με υστέρηση (hysteresis thresholding) των τοπικών μεγίστων που εντόπισε το προηγούμενο βήμα [42]. Για αυτό το βήμα, ανατίθεται κατώτατο και ανώτατο όριο κατωφλίωσης. Όσες ακμές έχουν διεύθυνση κλίσης μεγαλύτερη από το ανώτατο όριο είναι σίγουρα ακμές ενώ όσες είναι μικρότερες από το κατώτατο αφαιρούνται από το σύνολο των ακμών. Όσες είναι εντός των ορίων, θεωρούνται ακμές μόνο αν κάποιο γειτονικό τους εικονοστοιχείο είναι σίγουρη ακμή αλλιώς απορρίπτονται. Έτσι, απομένουν μόνο οι ισχυρές ακμές και μειώνεται ο θόρυβος στη δυαδική φωτογραφία που εξάγεται [Εικόνα 2.7h].

2.3.2 Ανάλυση Εικόνας

Λόγω της αλληλοεπικάλυψης σε διάφορα σημεία με τον όρο «ανάλυση εικόνας», κατά την εργασία ως επεξεργασία εικόνας θα ορίζεται η διαδικασία που έχει σαν είσοδο και έξοδο εικόνα. Το κομμάτι της ανάλυσης εικόνων έχει αυστηρές εισόδους καθώς πρέπει να γίνει κατάλληλη επεξεργασία της ψηφιακής εικόνας πριν είναι δυνατή η εξαγωγή δεδομένων [Εικόνα 2.9][43]. Κατά την ανάλυση εικόνας, αξιοποιούνται τα αντικείμενα που έχουν γίνει διακριτά δια μέσω της επεξεργασίας και προσδίδεται η συμβολική τους αξία. Για την παραγωγή συμβολικών δεδομένων, αλλά και για την ερμηνεία τους γίνεται χρήση μοντέλων που δομούνται στρατηγικά με διεπιστημονική μελέτη. Η συμβολή της γεωμετρίας, της φυσικής και της στατιστικής είναι απαραίτητη στην πλειοψηφία των προβλημάτων που καλείται να λύσει η μηχανική όραση [44].



Εικόνα 2.9: Ανάλυση Εικόνας με εξαγωγή και αναγνώριση αντικειμένων [45].

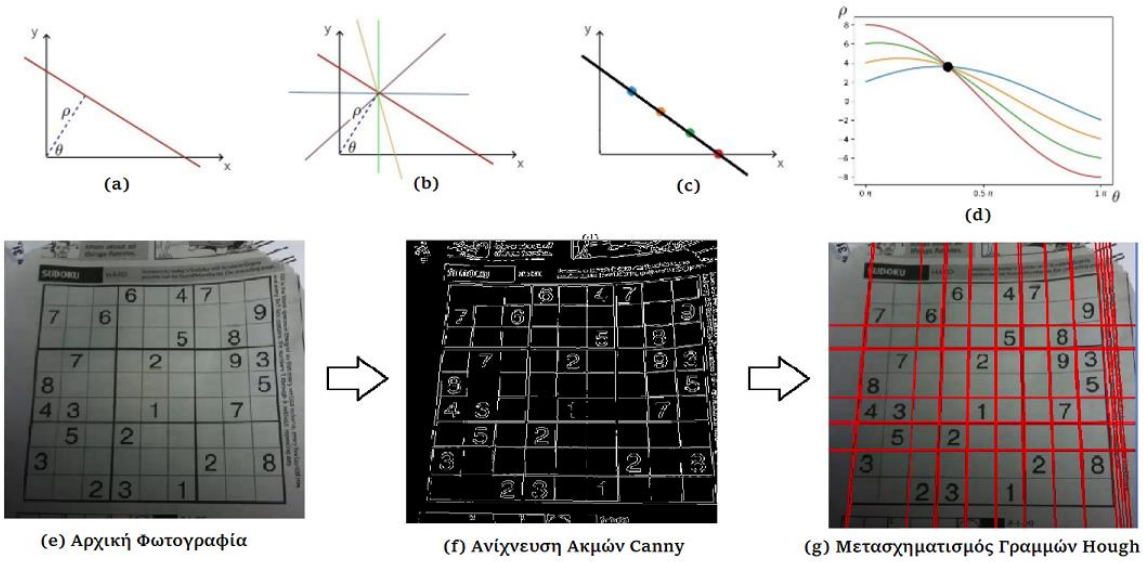
2.3.2.1 Μετασχηματισμός Γραμμών Hough

Αξιοποιώντας τις ακμές που εξάγει μία συνάρτηση όπως η Ανίχνευση Ακμών Canny, ο Μετασχηματισμός Γραμμών Hough μπορεί να εντοπίσει ευθείες γραμμές [46]. Μία γραμμή μπορεί να εκφραστεί από δύο μεταβλητές, είτε ως (a, b) στο Καρτεσιανό σύστημα συντεταγμένων με έκφραση της γραμμής ως $y = ax + b$ **(2.5)** είτε ως (ρ, θ) με έκφραση $\rho = xcos(\theta) + ysin(\theta)$ **(2. 6).** Για τη δεύτερη περίπτωση, το ρ είναι η κάθετη απόσταση από την αρχή των αξόνων έως τη γραμμή ενώ το θ η γωνία που σχηματίζει η κάθετη γραμμή της απόστασης και ο οριζόντιος άξονας αριστερόστροφα [Εικόνα 2.10a].

Για κάθε σημείο (x_0, y_0) υπάρχει ένα σύνολο γραμμών που διέρχονται από το συγκεκριμένο σημείο με τη μορφή $\rho = x_0 cos(\theta) + y_0 sin(\theta)$ (εφαρμογή της εξίσωσης 2.6). Όπως είναι εμφανές, κάθε γραμμή που διέρχεται από το (x_0, y_0) μπορεί να εκφραστεί από το ζεύγος $(\rho, \theta),\ για\ κάθε\ \rho > 0\ και\ 0 < \theta < 2\pi$ [Εικόνα 2.10b]. Σχεδιάζοντας το διάγραμμα

του συνόλου γραμμών που διέρχονται από ένα ζεύγος (x_0, y_0) προκύπτει μία ημιτονοειδής συνάρτηση.

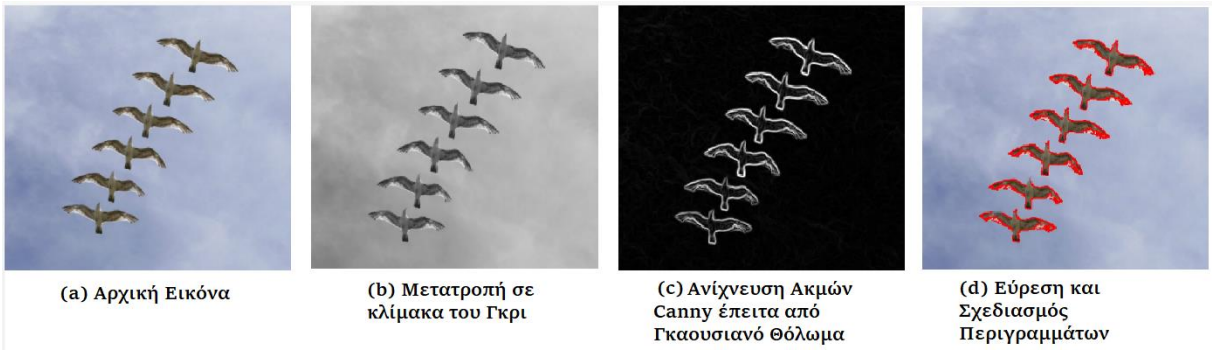
Κάνοντας αυτή τη διαδικασία για κάθε ακμή της ψηφιακής φωτογραφίας δίνεται η δυνατότητα να εντοπιστούν ποιες ανήκουν πάνω στην ίδια γραμμή. Όταν δύο ή περισσότερες ημιτονοειδείς συναρτήσεις τέμνονται στο ίδιο σημείο σημαίνει πως τα σημεία που εκφράζουν βρίσκονται πάνω στην ίδια γραμμή [Εικόνα 2.10c-d]. Η διαδικασία εντοπισμού γραμμών Hough και ο σχεδιασμός τους εξετάζεται στις Εικόνες 2.10e-g. Ο αλγόριθμος κατατάσσεται ως μέθοδος ανάλυσης εικόνας καθώς δεν έχει ψηφιακή φωτογραφία ως έξοδο, αλλά ζεύγη (ρ, θ) που μετέπειτα μπορούν να σχεδιαστούν πάνω σε μία εικόνα.



Εικόνα 2.10: Μετασχηματισμός Γραμμών Hough a-d: [Hough Lines Transform Explained] e-g:[OpenCv2].

2.3.2.2 Εύρεση Περιγραμμάτων

Μία από τις πιο βασικές πτυχές της ανάλυσης εικόνων είναι η εύρεση περιγραμμάτων (contours) σε μία ψηφιακή εικόνα [47]. Τα περιγράμματα είναι καμπύλες που ενώνουν γειτονικά σημεία που έχουν ίδιες ιδιότητες όπως χρώμα ή ένταση. Ιδιαίτερη σημασία για την αποτελεσματική εύρεση των περιγραμμάτων έχει η κατάλληλη επεξεργασία της ψηφιακής φωτογραφίας. Συνήθως, καλύτερα αποτελέσματα προσδίδουν οι δυαδικές φωτογραφίες όπως αυτές που εξάγει η κατωφλίωση ή η ανίχνευση ακμών Canny. Μέσω των περιγραμμάτων γίνεται δυνατή η εύρεση και αναγνώριση αντικειμένων ή σχημάτων όπως παρουσιάζεται στην Εικόνα 2.11. Οι τεχνικές εύρεσης περιγραμμάτων δεν εξάγουν εικόνα, αλλά αντικείμενα για χρήση από άλλες μεθόδους.



Εικόνα 2.11: Διαδικασία Εύρεσης και Σχεδιασμού Περιγραμμάτων (a):[Vikram B. Baliga].

2.3.2.3 Εξαγωγή Δεδομένων Περιγραμμάτων

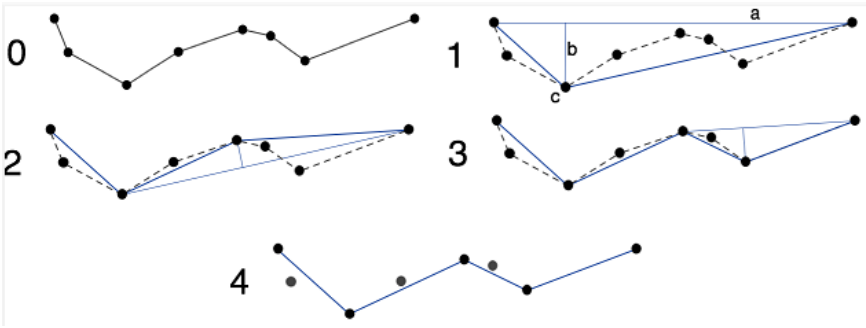
Τα αντικείμενα που εξάγονται από τις μεθόδους εύρεσης περιγραμμάτων κυρίως επεξεργάζονται δια μέσω των ιδιοτήτων τους. Με επιμέρους συναρτήσεις που αξιοποιούν τις ιδιότητες αυτές μπορούν σταδιακά να δομηθούν τα συμβολικά δεδομένα. Συναρτήσεις όπως υπολογισμός εμβαδού και περιμέτρου, οριοθέτησης και προσέγγισης του περιγράμματος είναι τα κύρια εργαλεία για κάθε διεργασία [47].

Μέσω του εμβαδού και της περιμέτρου, γίνεται δυνατό το φιλτράρισμα των περιγραμμάτων που δεν ανταποκρίνονται στις προδιαγραφές των αντικειμένων που θέλει να ομαδοποιήσει ο χρήστης. Φυσικά, το εύρος εμβαδού ή της περιμέτρου κρίνεται μέσα από πολλαπλές δοκιμές και μελέτη του αντικειμένου που θέλει να εντοπίσει.

Με την οριοθέτηση από ορθογώνιο ή έλλειψη, ο αλγόριθμος περικλείει το περίγραμμα συναρτήσει των συντεταγμένων ακραίων ακμών. Αυτό επιτρέπει στο χρήστη να απομονώσει τα πεδία ενδιαφέροντος για περαιτέρω επεξεργασία ή να κάνει υπολογισμούς με χρήση των συντεταγμένων διάφορων σημείων του περιγράμματος (όπως η εύρεση του κέντρου του περιγράμματος).

Η προσέγγιση του περιγράμματος είναι πιο σύνθετη συνάρτηση κατά την οποία ο αλγόριθμος προσπαθεί να προσεγγίσει το σχήμα του περιγράμματος χρησιμοποιώντας λιγότερα σημεία με σεβασμό τόσο στο αρχικό σχήμα όσο και στην ακρίβεια ε που θέτει ο χρήστης [Εικόνα 2.12]. Η προσέγγιση γίνεται μέσω του αλγορίθμου Ramer-Douglas-Peucker. Σύμφωνα με τον αλγόριθμο, η αρχική καμπύλη διαιρείται σε κομμάτια προς εξέταση. Για κάθε κομμάτι, μαρκάρεται το πρώτο και το τελευταίο σημείο προς διατήρηση. Έπειτα, βρίσκεται το σημείο με τη μεγαλύτερη κάθετη απόσταση από τη γραμμή που ενώνει αρχή και τέλος. Αν η απόσταση είναι μικρότερη από το ε το σημείο απορρίπτεται, ενώ αν είναι μεγαλύτερη το σημείο μαρκάρεται για να διατηρηθεί. Αφού εκτελεστεί για όλα τα κομμάτια

της αρχικής καμπύλης, εξάγεται η νέα καμπύλη που αποτελείται από όσα σημεία έχουν μαρκαριστεί προς διατήρηση.



Εικόνα 2.12: Αλγόριθμος Ramer-Douglas-Peucker [Ramer-Douglas-Peucker].

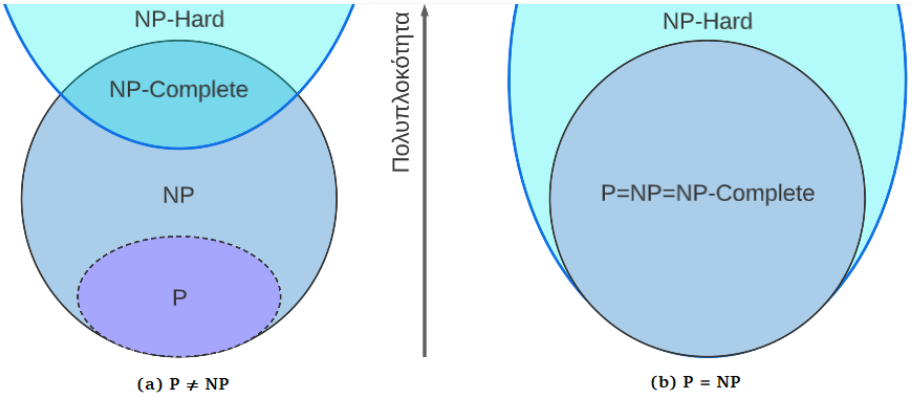
2.4 Κλάσεις Πολυπλοκότητας Προβλημάτων

Οι κλάσεις πολυπλοκότητας αποτελούν τον ακρογωνιαίο λίθο της θεωρίας πολυπλοκότητας και κεντρικό ζήτημα του κλάδου της θεωρητικής πληροφορικής. Κάθε κλάση περιέχει προβλήματα που απαιτούν ίδιο χώρο και χρόνο, ενώ συνήθως η ταξινόμηση τους γίνεται εκτελώντας το πρόβλημα σε ένα αυτόματο όπως οι Μηχανές Turing. Η ιεραρχία των κλάσεων δομείται με την οργάνωση τους σε σύνολα και υποσύνολα. Για παράδειγμα, η κλάση P που περιέχει τα προβλήματα που επιλύονται σε ντετερμινιστικό πολυωνυμικό χρόνο είναι υποσύνολο της κλάσης προβλημάτων NP που λύνονται σε μη ντετερμινιστικό πολυωνυμικό χρόνο [Εικόνα 2.13] [48], [49].

Η μελέτη των κλάσεων συνεισφέρει στην πληροφορική ταξινομώντας τα προβλήματα κατά το χώρο και το χρόνο που απαιτείται για την επίλυση και επαλήθευση της λύσης [50]. Η χρονική πολυπλοκότητα ενός αλγορίθμου περιγράφεται από τον αριθμό βημάτων που χρειάζονται για την επίλυση ενός προβλήματος, αλλά μπορεί να περιγράψει και πόσο χρόνο παίρνει η επιβεβαίωση της λύσης. Γνωρίζοντας την χρονική πολυπλοκότητα οι προγραμματιστές έχουν μία εικόνα για την απόδοση του αλγορίθμου τους. Η χωρική πολυπλοκότητα περιγράφει τη μνήμη που απαιτεί ο αλγόριθμος για την εκτέλεσή του.

Μερικά από τα πιο μεγάλα προβλήματα που δεν έχουν επιλυθεί πραγματεύονται την απόδειξη της ισότητας μεταξύ δύο υποσύνολων. Ένα από τα πιο γνωστά προβλήματα είναι το P = NP που αναφέρθηκε κατά την εισαγωγή της εργασίας [51], [52]. Η επίλυση του προβλήματος αυτού θα προκαλούσε χάος στην προγραμματιστική κοινότητα καθώς πολλοί

από τους αλγορίθμους ασφαλείας βασίζονται στο γεγονός ότι δεν υπάρχει παράκαμψη στις υπολογιστικές απαιτήσεις των προβλημάτων NP [Εικόνα 2.13b].



Εικόνα 2.13: Κλάσεις Πολυπλοκότητας (a) P≠NP (b) P=NP.

2.4.1 Κλάση NP

Η κλάση NP περιέχει προβλήματα απόφασης που λύνονται σε πολυωνυμικό χρόνο με μία μη-ντετερμινιστική μηχανή Turing [53]. Αν και η επίλυση τους μπορεί να απαιτήσει πολύ χρόνο, μπορεί να επαληθευτεί αν η λύση είναι σωστή από μία μηχανή Turing σε πολυωνυμικό χρόνο. Η επαλήθευση αυτή ονομάζεται "μάρτυρας" ή "πιστοποίηση" της επίλυσης και είναι απαραίτητη για να αποδειχθεί ότι ένα πρόβλημα ανήκει στην κλάση NP. Υποσύνολο της κλάσης NP είναι η κλάση P (P⊂NP) [Εικόνα 2.13a]. Γνωστά προβλήματα κλάσης NP είναι: ο ισομορφισμός και χρωματισμός γράφων [54], το πρόβλημα πλανόδιου πωλητή [55], η παραγοντοποίηση ακεραίων [56], SAT [57].

2.4.2 Κλάση P

Η κλάση P περιέχει προβλήματα απόφασης (προβλήματα που μπορούν να απαντηθούν με ναι ή όχι) που λύνονται σε πολυωνυμικό χρόνο με μία ντετερμινιστική μηχανή Turing [53]. Πολλά από τα προβλήματα της λύνονται με προσεγγίσεις brute-force (χρήση υπολογιστικής δύναμης δοκιμάζοντας κάθε πιθανό συνδυασμό για εύρεση λύσης) αλλά συχνά απαιτούν εκθετικό χρόνο. Φυσικά αν το πρόβλημα διαθέτει αλγόριθμο πολυωνυμικού χρόνου μπορεί να λυθεί πολύ πιο αποδοτικά. Ένα πρόβλημα κλάσης P λύνεται σε n^k χρόνο όπου k είναι σταθερά και n το μέγεθος της εισόδου [53]. Επιπρόσθετη βελτίωση μπορεί να γίνει με τη χρήση αλγορίθμου Karatsuba κατά τον οποίο συναντώνται μικρότερες

σταθερές [58]. Γνωστά προβλήματα κλάσης P είναι: ο γραμμικός προγραμματισμός [59], ο έλεγχος πρώτων αριθμός [60], η σύγκριση συμβολοσειρών, το ελάχιστο επικαλύπτων δένδρο.

2.4.3 Κλάση NP-Complete

Τα προβλήματα NP-Complete είναι ένα υποσύνολο προβλημάτων στα οποία μπορεί να αναχθεί σε πολυωνυμικό χρόνο καθένα από τα προβλήματα της NP και η λύση των οποίων μπορεί να επαληθευτεί σε επίσης πολυωνυμικό χρόνο [61]. Ως αποτέλεσμα του ορισμού, αν υπήρχε ένας αλγόριθμος πολυωνυμικού χρόνου για τη λύση NP-Complete προβλημάτων θα μπορούσε να λυθεί κάθε πρόβλημα της NP σε ανάλογο χρόνο. Το θεώρημα Cook-Levin δείχνει με τη μέθοδο απόδειξης δια αναγωγής πως τα Προβλήματα Ικανοποίησης Boolean ανήκουν στην κλάση NP-Complete [62].

Εκατοντάδες προβλήματα έχουν διατυπωθεί και εισαχθεί στην εκτενή λίστα της κλάσης NP-Complete. Σε τομείς όπως οι γράφοι, ο μαθηματικός προγραμματισμός, η ανάλυση κειμένου και συμβολοσειρών αλλά και στα παιχνίδια συναντώνται και εκφράζονται τέτοια προβλήματα. Μερικά γνωστά παραδείγματα αποτελούν: το πρόβλημα σακιδίου [63], ο προγραμματισμός ακεραίων, ο ισομορφισμός γράφων [54], ο χρωματισμός γράφων [64], το πρόβλημα μονοπατιού Hamilton [65] και η δρομολόγηση οχημάτων.

2.5 Μοντέλα Προβλημάτων

Κάνοντας αναγωγή του προβλήματος σε κυκλωματικό SAT ο Kaye απέδειξε πως ο Ναρκαλιευτής ανήκει στα προβλήματα NP-Complete [10]. Ορισμένες σύνθετες καταστάσεις του Ναρκαλιευτή μπορούν να διατυπωθούν ως μοντέλα CSP και να κωδικοποιηθούν σε μορφή Boolean. Αυτό επιτρέπει να λυθούν με εργαλεία που κάνουν συνδυασμό SAT και προγραμματισμού περιορισμών. Είναι κρίσιμο να αναλυθούν τα δύο μοντέλα προβλημάτων ώστε να μπορεί να περιγραφεί η διαδικασία.

2.5.1 CSP

Ως Προβλήματα Ικανοποίησης Περιορισμών (CSP) ορίζονται τα προβλήματα που η επίλυση τους περιλαμβάνει την ανάθεση κατάλληλων τιμών στις μεταβλητές τους ώστε να ικανοποιούν ορισμένους περιορισμούς ή κανόνες [66]. Συνήθως είναι απαραίτητη η χρήση ευριστικών μεθόδων και αλγόριθμων συνδυαστικής αναζήτησης για την επίλυσή τους σε

λογικό χρόνο καθώς τα CSP παρουσιάζουν μεγάλη χρονική πολυπλοκότητα. Με την εξαγωγή εφικτών λύσεων και τη βελτίωση των μεθόδων αυτών ασχολείται ο κλάδος του προγραμματισμού περιορισμών. Κατά την εργασία "A dichotomy theorem for nonuniform CSPs" [67], αποδεικνύεται πως ένα πεπερασμένο σύνολο προβλημάτων CSP λύνονται είτε σε πολυωνυμικό χρόνο (κλάσης P) είτε είναι NP-Complete.

Τα μοντέλα CSP ορίζονται με τρία βασικά κομμάτια $\langle X, D, C \rangle$:

- Σύνολο μεταβλητών $X = \{X1, X2, \ldots, XN\}$ **(2.7)**
- Σύνολο πεδίων ορισμού τιμών $D = \{D1, D2, \ldots, DN\}$ **(2.8)**
- Σύνολο Περιορισμών $C = \{C1, C2, \ldots, CN\}$ **(2.9)**

Σε κάθε μεταβλητή X_i ανατίθεται ένα πεδίο ορισμού D_i που δεν είναι κενό για να πάρει τις τιμές της. Κάθε περιορισμός C_i μπορεί να περιέχει οποιοδήποτε αριθμό μεταβλητών και παίρνει τιμές αληθές ή ψευδές αναλόγως με το αν ικανοποιείται.

$$C_k\colon powerset(D) \rightarrow \{true, false\}, k = 1,2, \ldots m \quad \textbf{(2.10)}$$

- Η ανάθεση είναι συνεπής αν τηρούνται όλοι οι περιορισμοί C.
- Η ανάθεση είναι πλήρης αν κάνει χρήση όλων των μεταβλητών X.
- Οι αναθέσεις μπορούν να είναι μερικές αν χρησιμοποιούν υποσύνολο της X.
- Μία ανάθεση αποτελεί λύση του προβλήματος ικανοποίησης περιορισμών αν είναι συνεπής και πλήρης.

Διάδοση Περιορισμών

Τα μοντέλα επίλυσης που βασίζονται στη διάδοση περιορισμών, μοντελοποιούν τον κάθε περιορισμό C ως μηχανισμό διάδοσης που περιορίζει το πεδίο ορισμού μιας μεταβλητής αφαιρώντας τις τιμές που δε συμμετέχουν σε καμία λύση λόγω ασυνέπειας. Το βασικό πλεονέκτημα αυτής της προσέγγισης είναι ότι οι μηχανισμοί διάδοσης κάθε περιορισμού μπορούν να δημιουργηθούν ανεξάρτητα και να χρησιμοποιηθούν σε συνδυασμό.

2.5.2 SAT

Τα Προβλήματα Ικανοποίησης Boolean (B-SAT ή πιο γνωστά ως SAT από τον όρο Satisfiability) είναι προβλήματα απόφασης που διατυπώνονται στη θεωρία πολυπλοκότητας. Αποτελούν πεδίο που αναλύει την επίλυση Προβλημάτων Ικανοποίησης Περιορισμών με

πεδία ορισμού μορφής Boolean και περιορισμούς ως λογικές πράξεις. Το SAT ήταν το πρώτο NP-Complete πρόβλημα που διατυπώθηκε από τον Stephen Cook το 1971 [62]. Με κατάλληλη κωδικοποίηση ενός CSP μπορεί να επιλυθεί ως SAT.

Κατά τα SAT προσδιορίζεται εάν υπάρχει ανάθεση των μεταβλητών X_i ενός προβλήματος αποκλειστικά με Boolean τιμές ($D_i = \{true, false\}$) η οποία να επαληθεύει πάντα την έκφραση που ορίζει το πρόβλημα. Ως κυριολεκτικό l_i (literal) ορίζεται η ανάθεση σταθερής τιμής μίας μεταβλητής X_i ή το συμπλήρωμά της ως $\neg l_i$. Μία πρόταση C αποτελείται από τη διάζευξη κυριολεκτικών $l_1 \vee l_2 \vee \ldots \vee l_n$ ενώ ένας τύπος Συζευκτικής Κανονικής Μορφής (CNF) από τη σύζευξη προτάσεων $C_1 \wedge C_2 \wedge \ldots \wedge C_n$. Ένας τύπος CNF ικανοποιείται όταν υπάρχει η ανάθεση κυριολεκτικών l που να τηρεί κάθε πρόταση C. Κάθε έκφραση Boolean μπορεί να διατυπωθεί ως CNF για τη μείωση του χώρου αναζήτησης [68].

Οι σύγχρονοι επιλυτές SAT [68] βασίζονται στη χρήση δυαδικών μεταβλητών (με τα αντίστοιχα κυριολεκτικά) και στους τύπους CNF. Κατά την αναζήτηση λύσης, δομείται σταδιακά μία μερική ανάθεση έως τη συμπλήρωση μίας πλήρης ανάθεσης. Σε κάθε βήμα πρέπει να τηρείται η συνέπεια ως προς κάθε πρόταση του τύπου CNF. Οι μηχανισμοί διάδοσης (και οι συνδυασμοί τους) είναι υπεύθυνοι για την τήρηση της συνέπειας αλλά και για την απλοποίηση του τύπου όπου είναι εφικτό. Έχουν μηχανισμό που ανιχνεύει συγκρούσεις (δηλαδή αναθέσεις όπου δεν τηρείται η συνέπεια), ώστε να αναλυθούν και να εφαρμόσουν αλγόριθμο backjumping υπαναχωρώντας ένα βήμα πριν την επιλογή της ανάθεσης που επέφερε τη σύγκρουση. Σε σύνθετα προβλήματα, οι συγκρούσεις είναι αναπόφευκτες με αποτέλεσμα, όταν αποθηκεύονται για μελλοντική χρήση, να αυξάνονται ραγδαία οι προτάσεις και να υποφέρει ο χρόνος εκτέλεσης.

Δεν υπάρχει γνωστός αλγόριθμος ο οποίος να λύνει κάθε SAT αποδοτικά και το ερώτημα του αν υπάρχει αλγόριθμος πολυωνυμικού χρόνου που να τα επιλύει είναι ανάλογο του προβλήματος P=NP που αναφέρθηκε κατά την εισαγωγή του κεφαλαίου. Ωστόσο, έχουν προγραμματιστεί αλγόριθμοι που να μπορούν να λύσουν απαιτητικά προβλήματα με δεκάδες χιλιάδες μεταβλητές και εκατομμύρια προτάσεις.

2.6 Variable-State Independent Decaying Sum

Ένας ευριστικός αλγόριθμος που μπορεί να εφαρμοστεί στην αναζήτηση ενός επιλυτή SAT είναι ο Variable-State Independent Decaying Sum (VSIDS) [69]. Ο αλγόριθμος ξεκινά

εξετάζοντας μικρά κομμάτια της έκφρασης CNF και αναθέτοντας μια βαθμολογία σε κάθε κυριολεκτικό l_i και στο συμπλήρωμά του $\neg l_i$. Η βαθμολογία είναι ανάλογη με τον αριθμό των προτάσεων C που χρησιμοποιούν την εκάστοτε μεταβλητή. Περιοδικά κατά την εκτέλεση και αναζήτηση περισσότερων κομματιών της έκφρασης, ο VSIDS διαιρεί όλες τις βαθμολογίες με μία σταθερά. Αυτό μεταφράζεται σε μείωση της επιρροής της παρουσίας κάποιας μεταβλητής σε προτάσεις που έχουν εξεταστεί νωρίτερα και σε ενδυνάμωση των βαθμολογιών με παρουσία σε πιο πρόσφατες προτάσεις. Ο VSIDS διαλέγει πάντα το κυριολεκτικό της μεταβλητής που έχει τη μεγαλύτερη βαθμολογία για να αποφασίσει που θα συνεχίσει την αναζήτηση.

Με οδηγό τις βαθμολογίες, το VSIDS μπορεί να εγγυηθεί πως κάθε ανάθεση τιμής σε μεταβλητή ικανοποιεί το μεγαλύτερο αριθμό των πρόσφατων προτάσεων της έκφρασης CNF. Δεδομένου ότι οι βαθμολογίες δεν επηρεάζονται από τις αναθέσεις, η υπαναχώρηση (backtracking) του επιλυτή σε περίπτωση ασυνέπειας είναι ακόμα πιο εύκολη.

2.7 Lazy Clause Generation

Το LCG είναι μια υβριδική τεχνική που ενισχύει έναν επιλυτή SAT με τεχνικές γραμμικού προγραμματισμού [Εικόνα 2.14] [70]–[72]. Συνδυάζει τα πλεονεκτήματα του προγραμματισμού περιορισμών πεπερασμένου πεδίου ορισμού (Finite Domain Propagation) με αυτά των επιλυτών SAT. Συγκεκριμένα, εκμεταλλεύεται τα μοντέλα υψηλού επιπέδου των FD καθώς και την αποδοτική αυτόνομη αναζήτηση των SAT με τη χρήση του VSIDS και nogoods (στοιχεία που καταγράφουν το λόγο κάποιας αποτυχίας). Ο επιλυτής LCG κωδικοποιεί τις ακέραιες μεταβλητές του προβλήματος CSP σε μεταβλητές τύπου Boolean και κάθε αλλαγή σε αυτές αντικατοπτρίζεται στα πεδία ορισμού τους. Θεωρητικά, αυτό σημαίνει πως το CSP μπορεί να λυθεί όπως τα SAT με τα ανάλογα υπολογιστικά προτερήματα.

Έστω ακέραιος x με αρχικό πεδίο ορισμού $a_0 \dots a_n$ ισχύει:

$$Ισότητα: [[x = y]], a_0 \leq y \leq a_n \qquad \textbf{(2.11)}$$

$$Ανισότητα: \big[[x \leq z]\big], a_0 \leq z < a_n - 1 \qquad \textbf{(2.12)}$$

Παράλληλα με τις μεταβλητές και για να μη γίνονται αχρείαστες αναθέσεις κωδικοποιούνται τα πεδία ορισμού και οι περιορισμοί σε προτάσεις με ανάλογες μετατροπές όπως:

$$\textit{Πεδίο Ορισμού: } \text{Έστω } D = \{1,2,3,4\} \rightarrow \{[[x \geq 1]], [[x \leq 4]]\} \quad \textbf{(2.13)}$$

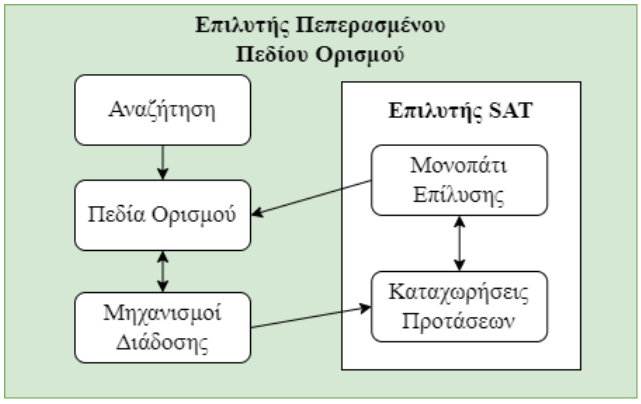
$$\textit{Περιορισμός: } C1 = [[x \leq z]] \rightarrow [[x \leq z + 1]] \quad \textbf{(2.14)}$$

$$\textit{Περιορισμός: } C2 = [[x = z]] \rightarrow [[x \leq z\ ]] \wedge \neg[[x \leq z - 1\ ]] \quad \textbf{(2.15)}$$

Οι ατομικοί περιορισμοί όπως η ανάθεση τιμής, οι αλλαγές πεδίων ορισμού ή η αφαίρεση τιμών από ένα πεδίο ορισμού είναι απλά κυριολεκτικά (literals) όπως στα SAT.

Η αναζήτηση ελέγχεται από έναν επιλυτή FD. Σε κάθε διάδοση περιορισμού, ο επιλυτής FD καταγράφει προτάσεις επεξήγησης και τις αποστέλλει στον εσωτερικό επιλυτή SAT ενεργοποιώντας τον με προτεραιότητα εκτέλεσης. Αφού ενεργοποιηθεί, δρα σαν καθολικός μηχανισμός διάδοσης με την υψηλότερη προτεραιότητα διάδοσης.

Όταν εκτελείται ένας μηχανισμός διάδοσης του επιλυτή FD για να τροποποιήσει το πεδίο ορισμού, δημιουργεί μία πρόταση επεξήγησης (explanation clause) κατά την οποία εξηγεί το λόγο που έγιναν αλλαγές σε ατομικούς περιορισμούς. Για παράδειγμα, η έκφραση $[x \leq 2] \wedge [x \geq y]$ σημαίνει πως $[y \leq 2]$ και δημιουργείται η πρόταση $[x \leq 2] \rightarrow [y \leq 2]$ ως δικαιολόγηση. Σε περίπτωση αποτυχίας, δηλαδή όταν ένα πεδίο ορισμού είναι κενό ύστερα από αλλαγή, ο μηχανισμός διάδοσης πρέπει να εξηγήσει και την αποτυχία. Οι προτάσεις επεξήγησης μπορούν να χρησιμοποιηθούν για τη δημιουργία nogoods τα οποία αξιοποιούνται εύκολα από τους μηχανισμούς διάδοσης του SAT. Τα nogoods επιτρέπουν την παράλειψη ενός μεγάλου υποσυνόλου του χώρου αναζήτησης ενώ παράλληλα επιτρέπουν τη σωστή επιλογή του επιπέδου που θα γίνει η (μη-χρονολογική) υπαναχώρηση. Οποιαδήποτε αλλαγή πραγματοποιεί ένας μηχανισμός διάδοσης μπορεί διαδοχικά να προκαλέσει την ενεργοποίηση άλλου μηχανισμού, με αυτόν τον κύκλο να εκτελείται έως ότου να φτάσει σε ένα ορισμένο σημείο.



Εικόνα 2.14: Αρχιτεκτονική του LCG.

2.8 Πλατφόρμα Ανάπτυξης και Εργαλεία

2.8.1 C-Sharp (C#)

Η C-Sharp [73] είναι μια γλώσσα προγραμματισμού υψηλού επιπέδου η οποία αναπτύχθηκε από τη Microsoft στα πλαίσια της πλατφόρμας .NET [74]. Η πρώτη έκδοσή της κυκλοφόρησε τον Ιανουάριο του 2002 με στόχο το συνδυασμό της υπολογιστικής δύναμης της C++ με την προγραμματιστική ευκολία της Visual Basic και τις λειτουργίες της Java. Η C# είναι αντικειμενοστραφής γλώσσα που απλοποιεί την πρόσβαση σε μεθόδους ή αντικείμενα με τη γλώσσα σήμανσης XML (eXtensible Markup Language) και του πρωτόκολλου SOAP (Simple Object Access Protocol). Οι εκτενείς της βιβλιοθήκες και η επεκτασιμότητα μέσω του .NET την κάνουν μια ισχυρή αλλά και ευέλικτη γλώσσα.

2.8.2 Visual Studio

Με την ανάπτυξη του Visual Studio η Microsoft παρείχε ένα ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) για να αξιοποιηθούν στο έπακρο και οι 36 υποστηριζόμενες γλώσσες. Το Visual Studio χρησιμοποιείται για την ανάπτυξη εφαρμογών για υπολογιστή ή κινητό, ιστοσελίδων καθώς και εφαρμογών ιστού. Οι κύριες λειτουργίες του είναι ο επεξεργαστής κώδικα, ο αποσφαλματωτής (Debugger), ο οπτικός σχεδιαστής και ο διαχειριστής επεκτάσεων. Ο επεξεργαστής κώδικα περιέχει επισήμανση σύνταξης και «έξυπνη» αυτόματη συμπλήρωση κώδικα με τη χρήση του IntelliSense. Ο αποσφαλματωτής βοηθάει καταγράφοντας την εσωτερική αποτύπωση της μνήμης κατά την εκτέλεση κάθε βήματος της διεργασίας για την εύρεση σφαλμάτων. Ο οπτικός σχεδιαστής, όπως ο Windows Forms Designer, χρησιμοποιείται για την ανάπτυξη εφαρμογών με γραφική διεπαφή σχεδιάζοντας τη διάταξη τους χωρίς να χρειάζεται κώδικας για κάθε δομικό στοιχείο. Τέλος, ο διαχειριστής επεκτάσεων δίνει πρόσβαση στην αναζήτηση και την προσθήκη βιβλιοθηκών ή πακέτων επέκτασης από άλλους χρήστες.

2.8.3 OpenCV και OpenCVSharp

Το OpenCV (Open Source Computer Vision Library) είναι μια βιβλιοθήκη ανοιχτού κώδικα που παρέχει λύσεις μηχανικής όρασης [75]. Η ανάπτυξή της ξεκίνησε από την Intel

με κύριο στόχο να σημειωθεί πρόοδος στον τομέα της μηχανικής όρασης. Για να πραγματοποιηθεί αυτό, οι ερευνητές αποφάσισαν πως ο κώδικας πρέπει να είναι βελτιστοποιημένος, αλλά και διαθέσιμος δωρεάν σε κάθε πλατφόρμα.

Για τη C# είναι διαθέσιμο το περίβλημα (wrapper) του OpenCVSharp που προσφέρει συμβατότητα με την πλατφόρμα .NET. Μέσω του διαχειριστή επεκτάσεων, μπορεί να εγκατασταθεί η βιβλιοθήκη αυτή κάνοντας διαθέσιμες τις λειτουργίες που περιγράφηκαν κατά την ενότητα 2.4 για την ψηφιακή επεξεργασία και ανάλυση εικόνας με τη χρήση συναρτήσεων.

2.8.4 Χώρος ονομάτων System

Ένας από τους πολυάριθμους χώρους ονομάτων (namespaces) που βρίσκονται στον πυρήνα της C# είναι ο χώρος System. Περιέχει θεμελιώδεις κλάσεις που ορίζουν τύπους δεδομένων, συμβάντα με τους αντίστοιχους χειριστές τους και διεπαφές όπως τα Windows Forms.

- **System.Windows.Forms**

Ο χώρος ονομάτων System.Windows.Forms περιέχει κλάσεις για τη δημιουργία διεπαφών στο λειτουργικό σύστημα Microsoft Windows ενώ διευκολύνει τη διαχείριση και προσομοίωση συμβάντων ποντικιού όπως μετακίνηση και κλικ.

- **System.Drawing**

Ο χώρος ονομάτων System.Drawing προσφέρει πρόσβαση σε βασικές λειτουργίες γραφικών και διαχείρισης αρχείων εικόνων. Επιτρέπει την αποθήκευση ενός στιγμιότυπου της οθόνης με αρχή και διαστάσεις ορισμένες από δομές Point και Size. Η δομή Point αντιπροσωπεύει ένα ζεύγος ακέραιων συντεταγμένων (x, y) σε δισδιάστατο επίπεδο ενώ η δομή Size ένα ζεύγος ακέραιων (h, w) που αντιστοιχούν στο ύψος και το πλάτος.

- **System.Linq**

Η LINQ (Language INtegrated Query) είναι ένα πρόσθετο της .NET για το χειρισμό πλήθους δεδομένων με τη χρήση ερωτημάτων (queries) όπως γίνεται στη γλώσσα διαχείρισης βάσης δεδομένων SQL. Ο χώρος ονομάτων System.Linq περιέχει αυτό το πρόσθετο για την εξαγωγή αποτελεσμάτων με ερωτήσεις πάνω σε ένα σύνολο δεδομένων όπως ένας πίνακας ή μια λίστα.

- **System.Numerics**

Ο χώρος ονομάτων System.Numerics επεκτείνει τους βασικούς αριθμητικούς τύπους προσφέροντας σύνθετες δομές. Δομές όπως αυθαίρετα μεγάλοι προσημασμένοι ακέραιοι (BigInteger), μιγαδικοί αριθμοί και διανύσματα περιέχονται στο System.Numerics.

2.8.5 Google OR-Tools

Το OR-Tools είναι μία σουίτα ανοιχτού κώδικα που αναλαμβάνει την επίλυση προβλημάτων γραμμικού προγραμματισμού, προγραμματισμού περιορισμών, μεικτών ακεραίων, δρομολόγησης οχημάτων και των αντίστοιχων προβλημάτων βελτιστοποίησης. Αναπτύχθηκε το 2011 από τον Laurent Perron και βρήκε ιδιαίτερη επιτυχία στον παγκόσμιο διαγωνισμό προγραμματισμού περιορισμών MiniZinc Challenge κατακτώντας την πρώτη θέση σε διάφορες δοκιμασίες από το 2018 έως και το 2021 [76].

Για τη μοντελοποίηση και επίλυση ενός CSP μπορεί να γίνει χρήση του επιλυτή CP-SAT που προσφέρει το OR-Tools. Το CP-SAT χρησιμοποιεί έναν επιλυτή Lazy Clause Generation σε συνδυασμό με έναν επιλυτή SAT όπως περιγράφεται στην παρουσίαση "Search is Dead Long Live Proof" [72]. Η εκτέλεση του επιλυτή περιλαμβάνει ένα στάδιο προεπεξεργασίας πριν ξεκινήσει η επίλυση για τη μείωση του χώρου αναζήτησης. Στην αρχή της προεπεξεργασίας εξετάζονται όλοι οι περιορισμοί μέχρι να μην είναι δυνατές άλλες αλλαγές. Οι μεταβλητές δε διαγράφονται, αλλά τα πεδία ορισμού τους μπορεί να μειωθούν. Όσον αφορά τους περιορισμούς, μπορούν να σημανθούν ως κενοί αλλά δε μπορούν να διαγραφούν. Στο στάδιο αυτό, μπορούν να προστεθούν νέες μεταβλητές ή περιορισμοί μετά τους προϋπάρχοντες με την προϋπόθεση ότι οι περιορισμοί προστίθενται μόνο όταν χρειάζονται στο πρόβλημα χαρτογράφησης. Στο τέλος, όλες οι μεταβλητές αντιγράφονται στο μοντέλο χαρτογράφησης του προβλήματος αλλά αντιστοιχίζονται με κάποιο δείκτη μόνο αυτές που χρησιμοποιούνται σε κάποιο περιορισμό.

Αναλυτικά, η διαδικασία της προεπεξεργασίας περιλαμβάνει με τη σειρά:

1. Μείωση των πεδίων ορισμού και την απλοποίηση των περιορισμών
2. Επέκταση ή αποσύνθεση περιορισμών
3. Εντοπισμός ισοδυναμίας μεταβλητών και καθορισμός σχέσεων
4. Αντικατάσταση με κανονική αναπαράσταση
5. Αλλαγή μεταβλητών και έλεγχος διάδοσης

Η προεπεξεργασία παράγει δύο μοντέλα, το εσωτερικό μοντέλο για την επίλυση και ένα μοντέλο διοχέτευσης για τη συμπλήρωση της λύσης του αρχικού μοντέλου. Το LCG αναλαμβάνει κωδικοποιώντας το CSP σε μορφή Boolean και επιλύοντας ως SAT με τη μέθοδο που αναφέρθηκε στο κεφάλαιο 2.7.

2.9 Σύνοψη κεφαλαίου

Κατά το κεφάλαιο, παρουσιάστηκαν οι μέθοδοι που συνδυάστηκαν για την υλοποίηση της παρούσας διπλωματικής εργασίας. Έγινε μελέτη των κλάσεων πολυπλοκότητας αλλά και των θεωρητικών πτυχών των μοντέλων προβλημάτων που περιλαμβάνονται στην επίλυση του Ναρκαλιευτή. Στο κεφάλαιο που ακολουθεί, θα εξεταστεί η ενορχήστρωση των μεθόδων και αλγορίθμων που αναλύθηκαν για την επίλυση κάθε υποπροβλήματος που συναντάται σε μία παρτίδα.

Κεφάλαιο 3

Ανάλυση και Σχεδίαση Εφαρμογής

Σε αυτό το κεφάλαιο θα αναλυθεί η δόμηση του Αυτοματοποιημένου Επιλυτή από τις επιμέρους μεθόδους αναγνώρισης και επίλυσης καταστάσεων του ταμπλό. Παράλληλα, θα εξεταστεί η μοντελοποίηση σύνθετων καταστάσεων ως CSP και η εξαγωγή λύσεων μέσω ενός υβριδικού επιλυτή CP-SAT.

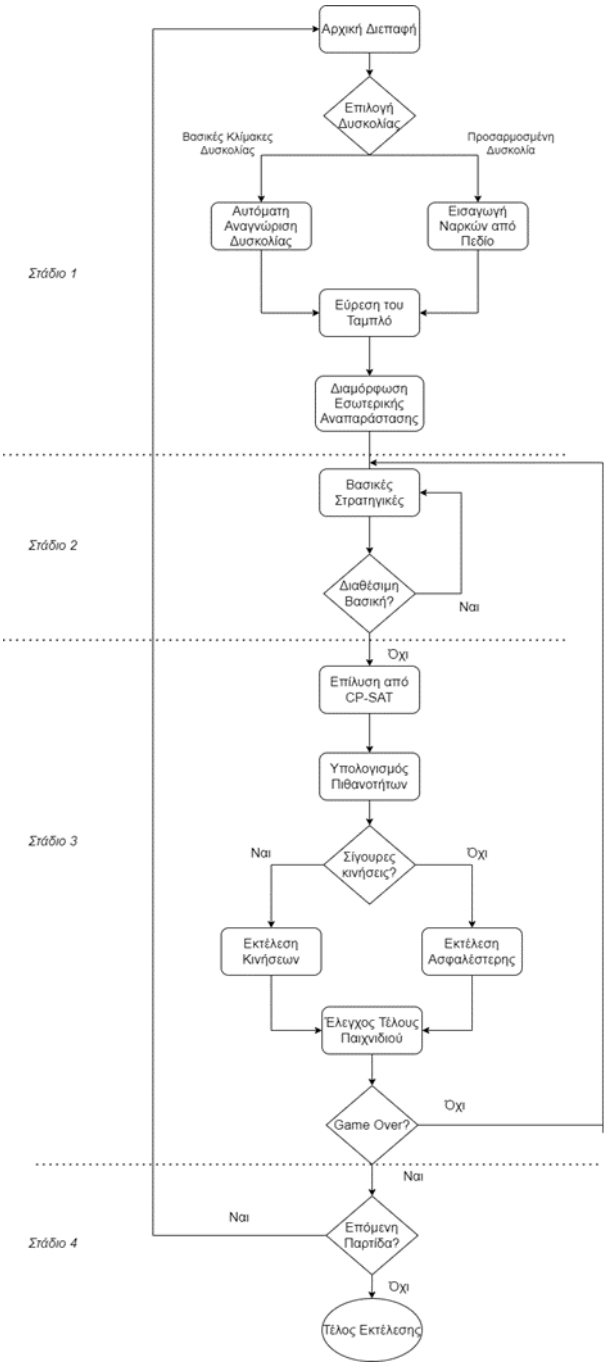
3.1 Γενική Επισκόπηση Επιλυτή

Ο Αυτόματος Επιλυτής αποτελείται από μία κεντρική διεπαφή για το χρήστη και τέσσερα στάδια εκτέλεσης όπως παρουσιάζονται στην Εικόνα 3.1. Ο χρήστης αποφασίζει πότε θέλει να ξεκινήσει η εκτέλεση και δεν απαιτείται άλλη δράση έως το τέλος της. Πριν πραγματοποιηθεί η προγραμματιστική υλοποίηση του επιλυτή, έγινε έρευνα για την επιλογή των αλγορίθμων που μπορούν να συνδυαστούν και να προσδώσουν ομαλή και αποδοτική εκτέλεση. Το μοντέλο ροής δεδομένων του Ναρκαλιευτή αναλύεται στην Εικόνα 3.2.

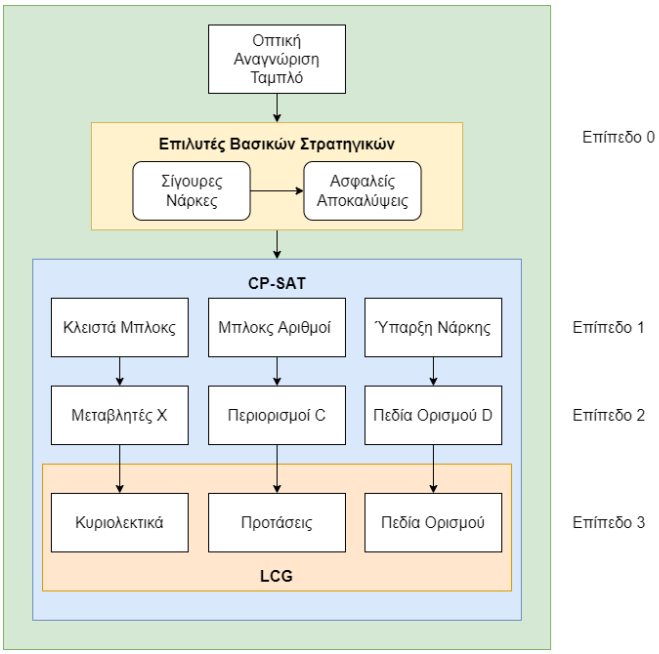
Για τη συγκρότηση της κάθε εισόδου και την εύρεση του ταμπλό του Ναρκαλιευτή, ο Αυτοματοποιημένος Επιλυτής της εργασίας αξιοποιεί μεθόδους μηχανικής όρασης με εφαρμογές ψηφιακής επεξεργασίας και ανάλυσης εικόνας. Αρχικά, καταγράφει ένα στιγμιότυπο της οθόνης, ώστε να βρεθεί το ταμπλό του παιχνιδιού και να διαμορφωθεί η εσωτερική αποτύπωση του. Αν ο χρήστης έχει επιλέξει τις βασικές κλίμακες δυσκολίας, ο επιλυτής αποφασίζει τον αριθμό ναρκών, ενώ αν επέλεξε προσαρμοσμένη δυσκολία οι νάρκες εισάγονται από αριθμητικό πεδίο. Σε κάθε κίνηση, γίνεται εκ νέου λήψη στιγμιότυπου για την αποτύπωση του αντίκτυπου της τελευταίας κίνησης στην εσωτερική αναπαράσταση. Εντοπίζονται τόσο οι αριθμοί (1-8) όσο και τα κλειστά μπλοκ για να αξιοποιηθούν σαν πληροφορίες στο επόμενο στάδιο.

Ο επιλυτής της εργασίας χρησιμοποιεί ντετερμινιστικές μεθόδους ως βασικές στρατηγικές. Για ορισμένες καταστάσεις ακολουθεί αποδοτικούς αλγορίθμους που εξάγουν απόλυτα ασφαλείς κινήσεις. Συγκεκριμένα, οι αλγόριθμοι βγάζουν συμπεράσματα που είναι σίγουρα νάρκες ή ασφαλείς αποκαλύψεις με βάση τα γειτονικά μπλοκ που περιβάλουν το εκάστοτε εξεταζόμενο μπλοκ. Για πιο απαιτητικές καταστάσεις που δεν καλύπτονται από το προηγούμενο στάδιο, εφαρμόζεται επίλυση ως Πρόβλημα Ικανοποίησης Περιορισμών με

κωδικοποίηση σε SAT. Αφού γίνει εξαγωγή των πιθανών αναθέσεων, ο επιλυτής υπολογίζει για κάθε μπλοκ την πιθανότητα να έχει νάρκη. Οι επιμέρους αλγόριθμοι στοχεύουν σε προβλήματα διαφορετικής πολυπλοκότητας οπότε είναι κρίσιμο να γίνεται χρήση των πιο σύνθετων στρατηγικών μόνο όταν είναι απαραίτητο. Στο τελικό στάδιο της εκτέλεσης, κρίνεται αν έχει επιτευχθεί ο στόχος με αναγνώριση των συνθηκών νίκης ή ήττας. Σε περίπτωση που η παρτίδα έχει ολοκληρωθεί, ο χρήστης μπορεί να επιλέξει αν θέλει να προχωρήσει σε επόμενη ή να τερματίσει την εκτέλεση.



Εικόνα 3.1: Διάγραμμα ροής του Αυτοματοποιημένου Επιλυτή με 4 στάδια – 1ο Στάδιο: Εύρεση του ταμπλό και αναγνώριση δυσκολίας, 2ο Στάδιο: Βασικές Στρατηγικές, 3ο Στάδιο Επίλυση ως CSP, 4ο Στάδιο: Τέλος Παιχνιδιού.

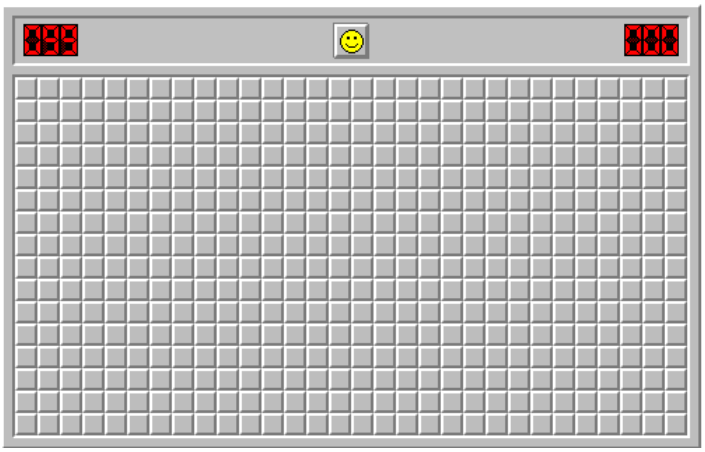


Εικόνα 3.2: Μοντέλο ροής δεδομένων- Επίπεδο 0: Διαδοχική εφαρμογή των βασικών στρατηγικών με οπτική αναγνώριση της κατάστασης του ταμπλό - Επίπεδο 1: Αρχική Είσοδος του CP-SAT - Επίπεδο 2: Μοντελοποίηση υψηλού επιπέδου ως CSP - Επίπεδο 3: Κωδικοποίηση ως Boolean για επίλυση ως SAT από το LCG.

3.2 Απαιτήσεις Εισόδου

Για την ορθή λειτουργία του αλγορίθμου απαιτείται η βασική έκδοση Windows XP του Ναρκαλιευτή δίχως τροποποιήσεις [Εικόνα 3.3] καθώς και η έκδοση 6.0 της δομής .NET. Σε αυτή την έκδοση του Ναρκαλιευτή τηρούνται οι διαστάσεις 16×16 εικονοστοιχείων για κάθε μπλοκ ενώ οι συνδυασμοί χρωμάτων έχουν μελετηθεί εις βάθος για την καλύτερη δυνατή ακρίβεια στο μοντέλο αναγνώρισης. Ο χρήστης πρέπει να έχει εξασφαλίσει την πρόσβαση στο παιχνίδι μέσω λογισμικού ή ιστοσελίδας που παρέχει τη συγκεκριμένη έκδοση.

Στην αρχή της εκτέλεσης, ο χρήστης πρέπει να επιλέξει μεταξύ των βασικών κλιμάκων δυσκολίας και της προσαρμοσμένης δυσκολίας. Αν ο χρήστης επιλέξει την προσαρμοσμένη δυσκολία, απαιτείται να εισάγει τον αριθμό των ναρκών στο αντίστοιχο πεδίο. Στο τέλος της παρτίδας, απαιτείται δράση για τον τερματισμό της εκτέλεσης ή την επανεκκίνηση του επιλυτή. Η διεπαφή και οι προτροπές προς το χρήστη θα παρουσιαστούν στην ενότητα που ακολουθεί.



Εικόνα 3. 3: Έκδοση Windows XP του Ναρκαλιευτή.

3.3 Διεπαφή Χρήστη

Για τη διευκόλυνση του χρήστη, ο Αυτοματοποιημένος Επιλυτής παρέχει μια γραφική διεπαφή (GUI) για τη διαχείριση του επιλυτή [Εικόνα 3.4]. Ο χρήστης μπορεί να επιλέξει «Base Game» για να ξεκινήσει μία παρτίδα στις βασικές κλίμακες δυσκολίας του Ναρκαλιευτή, ενώ για προσαρμοσμένες δυσκολίες μπορεί να επιλέξει «Custom Difficulty». Στη δεύτερη περίπτωση υπάρχει υπόδειξη για την εισαγωγή των ναρκών στο αριθμητικό πεδίο που ακολουθεί. Στο κάτω μέρος του παραθύρου, δίνεται η επιλογή τερματισμού της εκτέλεσης με το κουμπί «Exit Solver» ενώ με το κουμπί «About» γίνεται αναφορά στους παράγοντες που έκαναν δυνατή την υλοποίηση καθώς και τα πλαίσια της.

Εικόνα 3.4: Η αρχική οθόνη της διεπαφής του χρήστη με τις βασικές λειτουργίες της εκτέλεσης.

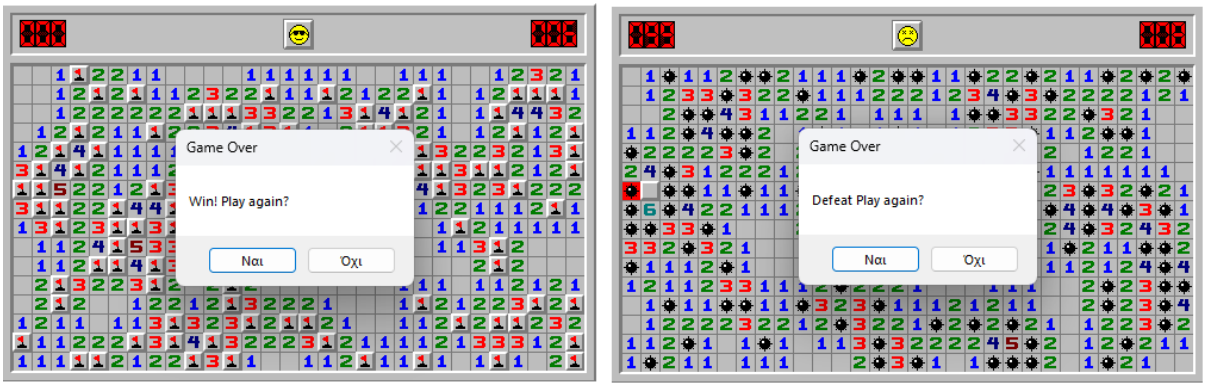
Μετά την εκκίνηση, υπάρχουν δύο πιθανές καταστάσεις για τον επιλυτή. Η κατάσταση που αποτυπώνεται στην Εικόνα 3.5a εμφανίζεται όσο η παρτίδα εξελίσσεται και ο επιλυτής διαμορφώνει κινήσεις. Η κατάσταση της εικόνας 3.5b παρουσιάζεται στο τέλος της

παρτίδας αφού ο μηχανισμός εύρεσης συνθηκών τερματισμού σημάνει πως το παιχνίδι έληξε ως νίκη ή ήττα.

(a) (b)

Εικόνα 3.5: Οι καταστάσεις του Αυτοματοποιημένου Επιλυτή – (a) Ο επιλυτής βρίσκεται εν ενεργεία (b) Ο επιλυτής ολοκλήρωσε την παρτίδα.

Κατά την κατάσταση της Εικόνας 3.5b, εμφανίζεται προτροπή προς το χρήστη για την επιλογή αν θέλει να προχωρήσει σε επόμενη παρτίδα ή να τερματίσει την εκτέλεση με το κατάλληλο μήνυμα αναλόγως την έκβαση της επίλυσης [Εικόνα 3.6]. Αν επιλέξει πως θέλει να επιλύσει επόμενη παρτίδα, ο επιλυτής επαναφέρει το ταμπλό στην αρχική του κατάσταση και εμφανίζεται και πάλι η αρχική διεπαφή.



(a) (b)

Εικόνα 3.6: Προτροπές προς το χρήστη κατά τη λήξη της παρτίδας – (a) Το μήνυμα σε περίπτωση νίκης (b) Το μήνυμα σε περίπτωση ήττας.

3.4 Εύρεση του Ταμπλό και Αναγνώριση Δυσκολίας

Ξεκινώντας την εκτέλεση του Αυτοματοποιημένου Επιλυτή, η βασικότερη προϋπόθεση που πρέπει να καλυφθεί είναι η εύρεση του ταμπλό. Για να λυθεί το πρόβλημα

της αναγνώρισης του ταμπλό [Αλγόριθμος 3.1] εφαρμόστηκαν μέθοδοι ψηφιακής επεξεργασίας και ανάλυσης εικόνων πάνω σε στιγμιότυπα που ελήφθησαν με συναρτήσεις για το χειρισμό του προσαρμογέα οθόνης. Ως έξοδος του αλγορίθμου είναι το παραλληλόγραμμο του ταμπλό με τις συντεταγμένες του πάνω στην οθόνη του χρήστη.

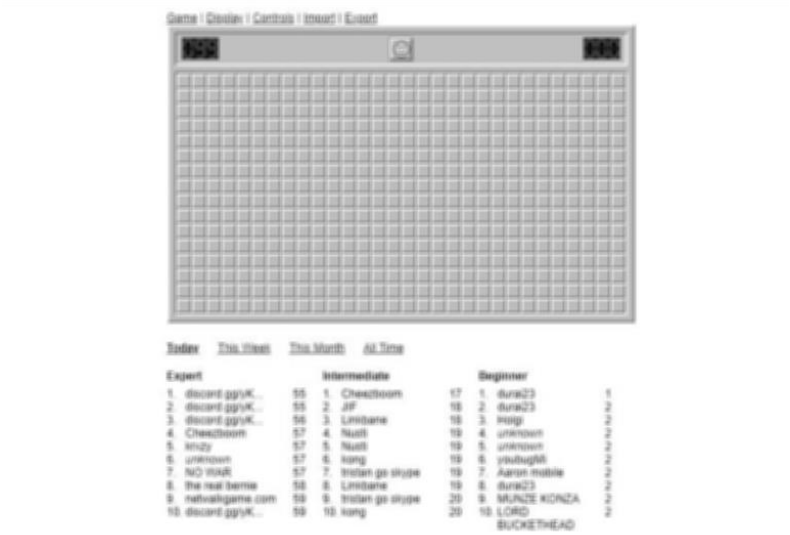
Algorithm 1: Find Game Board

Output: Board Rectangle Dimensions and Upper Left Corner (x,y)
img ← *screenshot*();
grayscale(*img*);
gaussianblur(*img*, (5, 5));
img → *fx*, *fy*;
edges ← *Canny*(*fx*, *fy*);
lines ← *HoughTransf*(*edges*);
if *lines* > *0* **then** –
  find most repeated minx,miny from P1;
  find most repeated maxx,maxy from P2;
else
  showerror();
return board;

Αλγόριθμος 3.1: Εύρεση του ταμπλό με συνδυασμό τεχνικών επεξεργασίας και ανάλυσης εικόνας.

Η συνάρτηση Find_Board ξεκινάει με τη λήψη στιγμιότυπου οθόνης μορφής Bitmap που διευκολύνει την ψηφιακή επεξεργασία. Ο αλγόριθμος εντοπίζει τις διαστάσεις της οθόνης και περικόπτει 80 εικονοστοιχεία από τα άκρα της οθόνης, εκεί που συνήθως βρίσκονται οι μπάρες εργαλείων. Παράλληλα, ελαχιστοποιεί τη γραφική διεπαφή δίνοντας ένα εύλογο χρονικό διάστημα (300ms) πριν τη λήψη, ώστε να μην επικαλύπτει το ταμπλό του Ναρκαλιευτή. Όπως αναλύθηκε στο προηγούμενο κεφάλαιο, υπάρχουν μέθοδοι που μπορούν να συνδυαστούν για την εύρεση ακμών και κατά επέκταση γραμμών όπως αυτές που σχηματίζουν το ταμπλό του Ναρκαλιευτή. Οι συναρτήσεις αυτές προσφέρονται από τη βιβλιοθήκη OpenCVSharp [75] η οποία αξιοποιείται εκτενώς στην εργασία για κάθε διεργασία που απαιτεί εργαλεία μηχανικής όρασης.

Το στιγμιότυπο του πρώτου βήματος φορτώνεται στη μνήμη με πρώτο στόχο την εύρεση ακμών. Για την εύρεση ακμών θα χρησιμοποιηθεί ο αλγόριθμος ανίχνευσης ακμών Canny [42] ο οποίος απαιτεί την προεπεξεργασία της ψηφιακής εικόνας με κατάλληλο τρόπο. Η εικόνα δέχεται αλλαγή χρωματικού μοντέλου [39] από BGR (Blue Green Red) σε κλίμακα του γκρι και έπειτα εφαρμόζεται Γκαουσιανό θόλωμα πλέγματος 5×5, όπως παρουσιάζεται στην εικόνα Εικόνα 3.7.



Εικόνα 3.7: Εφαρμογή αλλαγής χρωματικού μοντέλου και γκαουσιανού θολώματος πλέγματος 5x5.

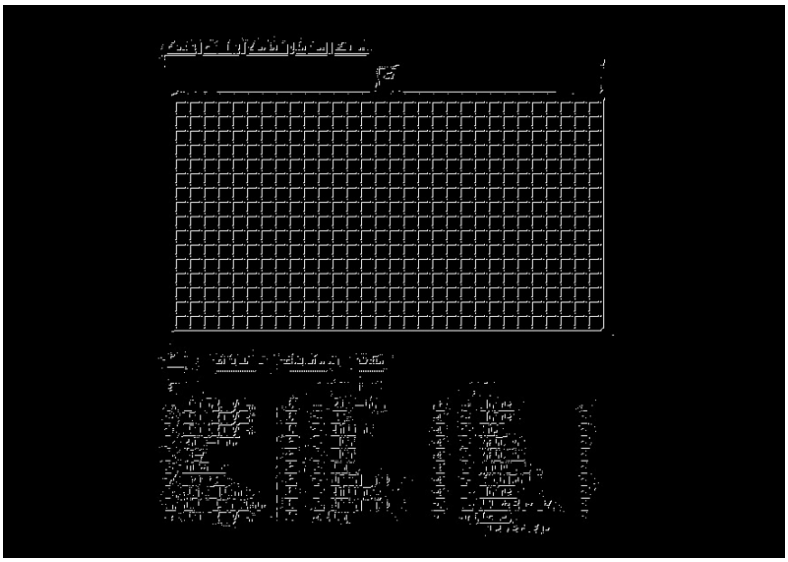
Έχοντας εφαρμόσει την προαπαιτούμενη προεπεξεργασία, πρέπει να καλυφθεί η ορθή μορφή εισόδου της συνάρτησης Canny. Η συνάρτηση απαιτεί ως είσοδο την επεξεργασμένη εικόνα σε μορφή CV_8U ή τις δύο συνιστώσες της (x και y) σε μορφή CV_16SC3 [42]. Οι μορφές αυτές υποδηλώνουν το πώς εκφράζεται η τιμή κάθε εικονοστοιχείου της ψηφιακής εικόνας. Ο γενικός τύπος και οι επιμέρους περιπτώσεις περιγράφονται στους παρακάτω τύπους:

$$CV\_ < βάθος\ bits > \{Unsigned|Signed|Float\}\ C < αριθμός\ καναλιών > \quad \textbf{(3.1)}$$

$$CV\_8U \rightarrow βάθος\ 8\ bits, U\ μη\ προσημασμένο, 1\ κανάλι \quad \textbf{(3.2)}$$

$$CV\_16SC3 \rightarrow βάθος\ 16\ bits,\ S\ προσημασμένο, C3\text{: } 3\ κανάλια \quad \textbf{(3.3)}$$

Στην περίπτωση του Αυτοματοποιημένου Επιλυτή, γίνεται χρήση των συνιστωσών της προεπεξεργασμένης εικόνας με τη μετατροπή από το αρχικό CV_8U σε CV_16SC3. Ύστερα από τη διαμόρφωση της εισόδου, ο αλγόριθμος Canny είναι έτοιμος να εκτελεστεί αφού τεθεί το πεδίο για την κατωφλίωση με υστέρηση. Καθώς οι τιμές της κατωφλίωσης αφορούν την αποτύπωση της εικόνας μετά το φίλτρο Sobel, μπορούν να είναι πάνω από το 255. Για την εξαγωγή του χάρτη ακμών της Εικόνας 3.8 εφαρμόστηκε πεδίο κατωφλίωσης 300 έως 400.

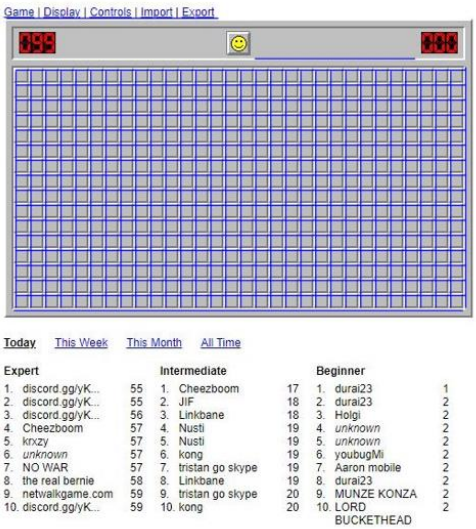


Εικόνα 3.8: Χάρτης Ακμών ως έξοδος του αλγορίθμου ανίχνευσης ακμών Canny με είσοδο δύο συνιστώσες της προεπεξεργασμένης εικόνας τύπου CV_16SC3 και πεδίο κατωφλίωσης υστέρησης 300-400.

Με την εξαγωγή του χάρτη ακμών, δίνεται η δυνατότητα εφαρμογής του αλγορίθμου μετασχηματισμού γραμμών Hough [46] για να βρεθούν τα σημεία που αποτελούν γραμμή. Ο αλγόριθμος αυτός απαιτεί σαν είσοδο το χάρτη ακμών, τις εκφράσεις του ζεύγους ρ και θ σε εικονοστοιχεία και ακτίνια αντίστοιχα καθώς και ένα κατώτατο όριο σημείων που αποτελούν μία γραμμή. Για επιπρόσθετη ακρίβεια, μπορούν να τεθούν σαν παράμετροι το ελάχιστο μήκος γραμμής και το μέγιστο κενό μεταξύ σημείων. Η εκτέλεση του αλγορίθμου για τον εντοπισμό των γραμμών του ταμπλό γίνεται με τις προδιαγραφές που περιγράφονται στον Πίνακα 3.1. Με βάση αυτές τις προδιαγραφές, η εκτέλεση εξετάζει διαδοχικά για κάθε ακμή του χάρτη ανά 1 μοίρα αν μπορούν να ενωθούν τουλάχιστον 75 σημεία για να σχηματίσουν μία γραμμή [Εικόνα 3.9].

Παράμετρος	**Τιμή**
ρ	1 εικονοστοιχείο
θ	$\frac{\pi}{180} rad = 1^o$
Κατώτατο όριο σημείων	75 ακμές του χάρτη
Ελάχιστο μήκος γραμμής	100 εικονοστοιχεία
Μέγιστο κενό μεταξύ σημείων	5 εικονοστοιχεία

Πίνακας 3.1: Οι παράμετροι εκτέλεσης του Μετασχηματισμού Γραμμών Hough για το ταμπλό του Ναρκαλιευτή.



Εικόνα 3. 9: Σχεδιασμός των γραμμών που εξάγονται από τον αλγόριθμο μετασχηματισμού Hough στην αρχική εικόνα με βάση τις παραμέτρους του Πίνακα 3.1 και το χάρτη ακμών της Εικόνας 3.8.

Η κάθε γραμμή που εξήγαγε ο μετασχηματισμός Hough αντιστοιχεί σε ένα σημείο $P1(x_1, y_1)$ και σε ένα σημείο $P2(x_2, y_2)$ που αποτελούν την αρχή και το τέλος της εκάστοτε γραμμής. Σε αυτό το στάδιο, ο σκοπός του αλγορίθμου είναι η εύρεση της πάνω αριστερής και κάτω δεξιάς γωνίας του ταμπλό ώστε να απομονωθεί. Βρίσκοντας αυτές τις δύο γωνίες μπορούν να υπολογιστούν οι διαστάσεις του ταμπλό και η τοποθεσία της αρχής του ταμπλό συναρτήσει των συνολικών διαστάσεων της οθόνης.

Οι γραμμές του ταμπλό μοιράζονται το ίδιο x_1 και x_2 ενώ οι στήλες μοιράζονται το ίδιο y_1 και y_2. Με τα δεδομένα αυτά, αρκεί να βρεθούν ποιες τιμές των x_2, y_2 είναι μέγιστες και ποιες τιμές των x_1, y_1 είναι ελάχιστες κατά επανάληψη για να εξακριβωθούν οι συντεταγμένες των δύο γωνιών. Αρχικά, επιλέγεται ένα τυχαίο σημείο P1 ως το σημείο με τις μικρότερες τιμές x_1 και y_1. Στη συνέχεια, εξετάζονται σταδιακά όλες οι γραμμές βρίσκοντας το ελάχιστο και το μέγιστο x και y που έχουν ταυτόχρονα τις περισσότερες εμφανίσεις στον πίνακα. Κάθε τιμή x και y έχει ένα τοπικό μετρητή για την προσμέτρηση των εμφανίσεών της. Για τη διασφάλιση ότι πρόκειται για το ταμπλό του παιχνιδιού, η πρώτη ανάθεση μεγίστων γίνεται μονάχα αν βρεθεί τουλάχιστον 4 φορές η εν λόγω τιμή του x_2, y_2 (επειδή οι μικρότερες δυνατές διαστάσεις του ταμπλό είναι 4×8). Οι μετέπειτα αναθέσεις μεγίστων και ελαχίστων γίνονται μονάχα αν ο τοπικός μετρητής είναι μεγαλύτερος από το μετρητή του τρέχοντος μεγίστου ή ελαχίστου.

$$Width = Round\left(\frac{\max x_2 - minx_1}{16}\right) \times 16\, pixels \quad \textbf{(3.4)}$$

$$Height = Round\left(\frac{\max y_2 - miny_1}{16}\right) \times 16\, pixels \quad \textbf{(3.5)}$$

$$Διαστάσεις\ Ταμπλό\ σε\ pixels \rightarrow Width \times Height \quad \textbf{(3.6)}$$

Ο υπολογισμός του μήκους και του ύψους που εκφράζουν τις διαστάσεις του ταμπλό σε εικονοστοιχεία παρουσιάζονται στις εξισώσεις 3.4-3.6. Δεδομένου ότι οι διαστάσεις του κάθε μπλοκ είναι 16×16, το μήκος και το ύψος πρέπει να είναι και αυτά γινόμενα του 16 για μέγιστη ακρίβεια. Παράλληλα, ως αρχή του ταμπλό στην οθόνη ανατίθεται η πάνω αριστερή γωνία δηλαδή το ζεύγος $(minx_1, miny_1)$. Ο καθορισμός της δυσκολίας για τις βασικές κλίμακες του παιχνιδιού γίνεται με βάση τις διαστάσεις του ταμπλό και τον πίνακα 2.1 που καταγράφει τις διαστάσεις (και τις ανάλογες νάρκες).

Όταν οι διαστάσεις του ταμπλό υπό εξέταση ταιριάζουν (έστω με απόκλιση κάποιων εικονοστοιχείων) με μία από τις κλίμακες δυσκολίας, ανατίθενται οι ανάλογες νάρκες στην κατάσταση του ταμπλό. Σε περίπτωση που ο χρήστης έχει επιλέξει προσαρμοσμένη δυσκολία, παραλείπεται ο έλεγχος αυτός και ανατίθενται οι νάρκες που έχει εισάγει ο ίδιος. Έπειτα από αυτή τη διαδικασία, δημιουργείται η εσωτερική αποτύπωση του ταμπλό. Σε κάθε μπλοκ θέτονται συντεταγμένες (x, y), κατάσταση, αριθμός ναρκών που αναζητεί, καθολική πιθανότητα καθώς και τοπικοί συνδυασμοί αναθέσεων ναρκών. Οι επιμέρους ιδιότητες θα εξεταστούν ενδελεχώς στις ενότητες που τις αξιοποιούν.

3.5 Ανάλυση Κατάστασης

Πριν από κάθε κίνηση του Αυτοματοποιημένου Επιλυτή πραγματοποιείται λήψη νέου στιγμιότυπου με χρήση του παραλληλόγραμμου που εξήγαγε η Εύρεση του ταμπλό. Πάνω σε αυτό το παραλληλόγραμμο εφαρμόζονται οι εξειδικευμένες μέθοδοι για την εύρεση της κατάστασης του κάθε μπλοκ και την ενημέρωση της εσωτερικής αναπαράστασης του ταμπλό. Κατά την πρώτη ανάλυση, όλα τα μπλοκ αναγνωρίζονται ως κλειστά σημάνοντας την αρχή της παρτίδας, ενώ στη συνέχεια αποτυπώνεται κάθε αλλαγή ενημερώνοντας το αρχικό μοντέλο. Η υλοποίηση αυτή συμφωνεί με τα πρότυπα του Packard [29] αλλά την επεκτείνει κάνοντας χρήση συμβολικών πληροφοριών αντί για φωτογραφίες. Κατά τη μέθοδο με τις φωτογραφίες, υπάρχει ένα σύνολο φωτογραφιών που καταγράφουν την κάθε κατάσταση που μπορεί να έχει ένα μπλοκ (κλειστό, ανοιχτό, 1-8, σημαία, νάρκη) και τις αναζητούν στο

σύνολο της οθόνης για να τις αντιστοιχίσουν με κάθε μπλοκ. Σε αντίθεση με τη μέθοδο αυτή, η υλοποίηση της παρούσας εργασίας κάνει χρήση μοντέλων για την αναγνώριση κάθε κατάστασης προσδίδοντας απόδοση σε μία από τις βασικότερες λειτουργίες για την επίλυση. Ο ψευδοκώδικας της αναγνώρισης της κατάστασης του ταμπλό αποδίδεται στον Αλγόριθμο 3.2.

Algorithm 2: Set Numbers and Closed Block

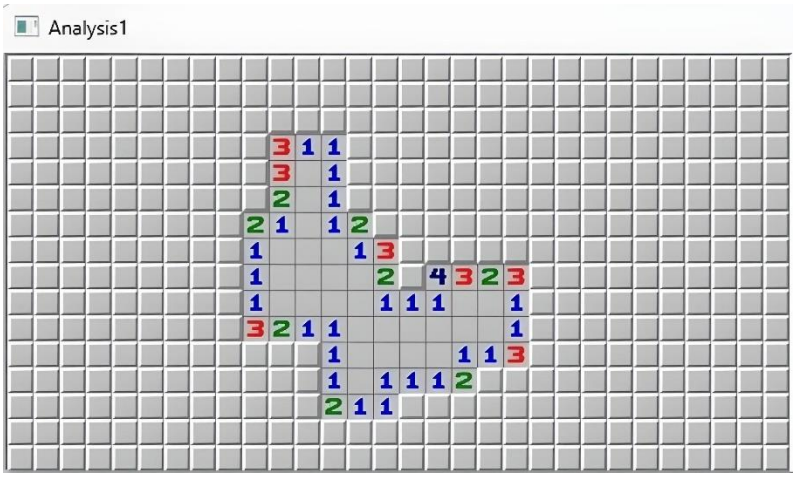
```
Output: Internal Interpration of the Board
Function FindNums(img):-
    colors ← setcolorranges();
    for j=0; j < colors.length; j++; do –
        inrangethresh(img, colors[j]);
        contours ← findcontours(img);
        for i=0; i < contours.length; i++; do –
            if contours.Area > 15 AND contours.Area< 50 then-
                index ← findindex(contours[i].Approximation());
                if numbers.Contains(index) then –
                    setnumber(index,j);
Function FindClosed(img):-
    colors ← setcolorranges();
    for j=0; j < colors.length; j++; do –
        inrangethresh(img, colors[j]);
        contours ← findcontours(img);
        for i=0; i < contours.length; i++; do –
            if contours.Area > 15 AND contours.Area< 50 then-
                index ← findindex(contours[i].Approximation());
                if numbers.Contains(index) then –
                    setnumber(index,j);
grayscale(img);
gaussianblur(img, (5, 5));
threshold(img, 100, 255);
contours ← findcontours(img);
for i=0; i < contours.length; i++; do –
    if contours.Area > 30 AND contours.Area< 120 then-
        numbers.Add(contours[i].BoundingRectangle());
FindNums(img);
FindClosed(img);
SetOpen(); // non-number or non-closed are set as open
return board;
```

Αλγόριθμος 3.2: Οπτική ανάλυση της κατάστασης του ταμπλό με τέσσερα στάδια – 1ο: Απομόνωση των αριθμών με ορθογώνια περιγράμματα και αποθήκευσή τους σε λίστα - 2ο: Εύρεση αριθμών και ανάθεση κατάστασης αν ανήκουν στη λίστα numbers - 3ο: Εύρεση κλειστών μπλοκ και ανάθεση κατάστασης – 4ο: Ανάθεση των υπολοίπων ως ανοικτά.

3.5.1 Εύρεση Αριθμών

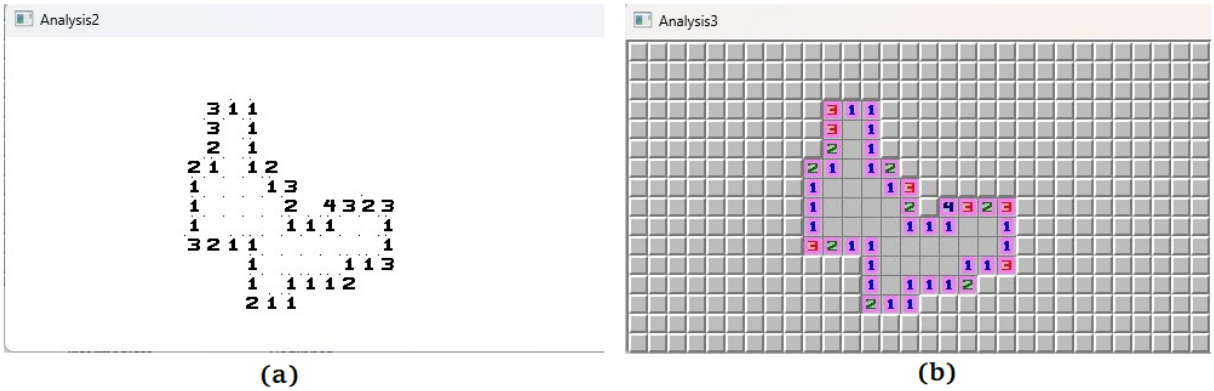
Αφού γίνει λήψη του στιγμιότυπου με προσέγγιση στις διαστάσεις του ταμπλό, καλείται άμεσα ο μηχανισμός εύρεσης αριθμών. Στη μέση κατάσταση (και ειδικά σε

κλίμακες μεγαλύτερης δυσκολίας) το ταμπλό αποτελείται από πολλαπλά διαφορετικά νούμερα προς εξέταση όπως στην Εικόνα 3.10.



Εικόνα 3.10: Ταμπλό του Ναρκαλιευτή με διαφορετικά νούμερα προς εξέταση για τη διαμόρφωση μοντέλου.

Για να προσδοθεί ακρίβεια στην αναγνώριση των αριθμών, εκτελείται μία συνάρτηση για να απομονωθούν μονάχα οι αριθμοί από το ταμπλό. Αρχικά, η Εικόνα 3.10 δέχεται αλλαγή χρωματικού μοντέλου σε κλίμακα του γκρι ενώ έπειτα εφαρμόζεται κατωφλίωση με εύρος 100 έως 255. Αυτό έχει ως αποτέλεσμα μία δυαδική εικόνα πάνω στην οποία μπορούν να γίνουν περαιτέρω δράσεις [Εικόνα 3.11]. Εισάγοντας την εικόνα 3.11a σε αλγόριθμο εύρεσης περιγραμμάτων, δημιουργείται ένας πίνακας με περιγράμματα. Με κατάλληλο φιλτράρισμα με βάση το εμβαδόν ($εμβαδόν > 30\ pixel\ και\ εμβαδόν < 120\ pixel$), παραμένουν μονάχα τα νούμερα. Στη συνέχεια, με συνάρτηση οριοθέτησης από ορθογώνιο επιστρέφεται και αποθηκεύεται σε λίστα ένα ορθογώνιο για κάθε νούμερο όπως αποτυπώνεται στην Εικόνα 3.11b. Με χρήση των συντεταγμένων (x, y), μπορούν να αποφευχθούν αποθηκεύσεις ορθογωνίων που υπάρχουν ήδη στη λίστα numbers.



Εικόνα 3.11: Διαδικασία απομόνωσης νούμερων – (a) Κατωφλίωση της Εικόνας 3.10 με εύρος 100-255.

Ακολουθώντας την διαδικασία απομόνωσης των νούμερων, καλείται η συνάρτηση colorise με στόχο την αναγνώριση κάθε αριθμού. Σε κάθε νούμερο αντιστοιχεί ένα εύρος τιμών έντασης BGR όπως αναγράφονται στον πίνακα 3.2. Υπάρχουν νούμερα, όπως το 1 και το 4, που έχουν αλληλεπικαλυπτόμενα εύρη τιμών έντασης στα εικονοστοιχεία από τα οποία αποτελούνται. Αυτό σημαίνει πως δίχως την απομόνωση, τα εύρη BGR θα έπρεπε να είναι αυστηρά για να υπάρχει ακρίβεια στην αναγνώριση. Δίνεται απόκλιση κάποιων μονάδων τόσο στο ανώτερο όσο στο κατώτερο όριο για κάθε αριθμό ώστε να εντοπίζονται όσο το δυνατόν περισσότερα εικονοστοιχεία.

Αριθμός	**Χρώμα**	**Κάτω Όριο (BGR)**	**Άνω Όριο (BGR)**
1	Μπλε Ανοιχτό	145,0,0	255,60,60
2	Πράσινο	0,60,0	60,255,60
3	Κόκκινο	0,0,160	60,60,255
4	Σκούρο Μπλε	4,0,0	124,30,30
5	Μπορντό	0,0,70	40,40,125
6	Γαλαζοπράσινο	80,90,2	130,135,50
7	Μαύρο	0,0,0	17,17,17
8	Σκούρο Γκρι	120,120,120	143,139,145

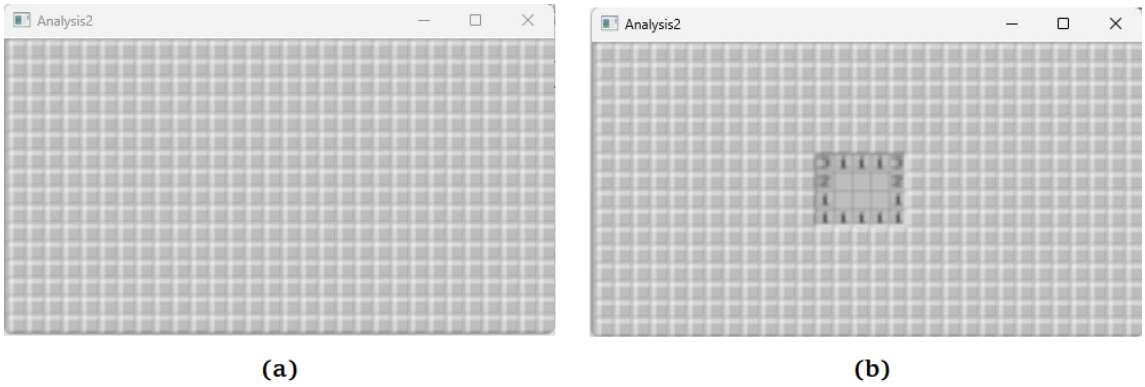
Πίνακας 3.2: Τα νούμερα του Ναρκαλιευτή και το εύρος τιμών έντασης BGR για την εύρεσή τους.

Η εύρεση των περιγραμμάτων των αριθμών γίνεται με συνάρτηση κατωφλίωσης εύρους με παραμέτρους το κατώτερο και ανώτερο όριο κάθε αριθμού. Διαδοχικά, εντοπίζεται το περίγραμμα των εικονοστοιχείων που είναι εντός εύρους και με κατάλληλο φιλτράρισμα με βάση το εμβαδόν διατηρείται το κύριο αντικείμενο. Στη συνέχεια, υπολογίζεται η περίμετρος του περιγράμματος ώστε να χρησιμοποιηθεί ως ακρίβεια για την προσέγγιση του περιγράμματος. Η συνάρτηση επιστρέφει το απλοποιημένο περίγραμμα από το οποίο μπορεί να βρεθεί με ακρίβεια ο δείκτης του μπλοκ στο οποίο ανήκει. Ο δείκτης υπολογίζεται με την παρακάτω εξίσωση για ένα τυχαίο σημείο (x_ι, y_ι) που ανήκει στο απλοποιημένο περίγραμμα:

$$Δείκτης\ Λίστας = Στήλες\ Ταμπλό \times \left(\frac{y_\iota}{16}\right) + \left(\frac{x_\iota}{16}\right) \quad \textbf{(3.7)}$$

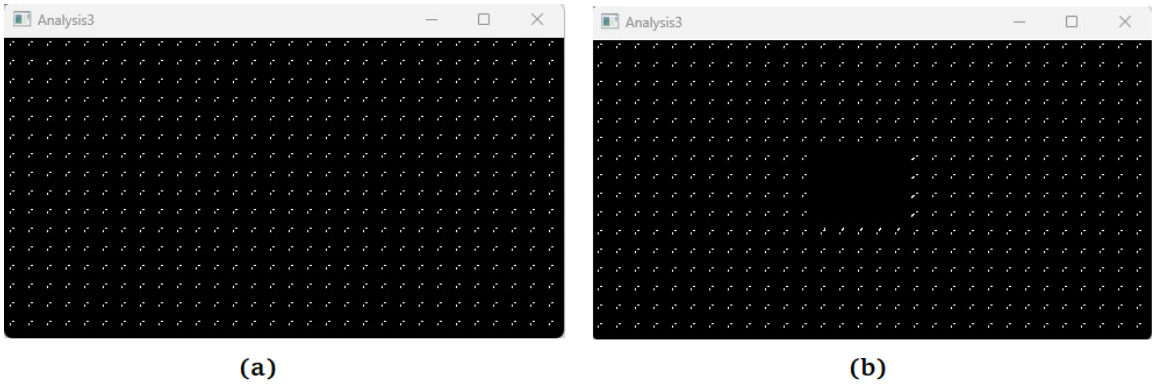
3.5.2 Εύρεση Ανοιχτών και Κλειστών Μπλοκ

Ύστερα από την εύρεση και αναγνώριση των αριθμών στο ταμπλό, τίθεται μία ακόμη πρόκληση με τη μορφή της εύρεσης των ανοιχτών και κλειστών μπλοκ. Η αναγνώριση των κλειστών μπλοκ είναι μια ιδιαίτερη περίπτωση καθώς τόσο τα ανοιχτά μπλοκ όσο και τα κλειστά έχουν αλληλοεπικάλυψη όσον αφορά το εύρος εντάσεων του γκρι. Με εκτενείς δοκιμές εντοπίστηκε ο κατάλληλος συνδυασμός τεχνικών για την εύρεση αξιόπιστης μεθόδου. Όπως παρουσιάζεται στην Εικόνα 3.12, στα πλαίσια της προεπεξεργασίας γίνεται εφαρμογή Γκαουσιανού θολώματος με πλέγμα 5×5, αλλαγή χρωματικού μοντέλου σε κλίμακες του γκρι καθώς και διαστολή της εικόνας.



Εικόνα 3.12: Καταστάσεις Ναρκαλιευτή ύστερα από Γκαουσιανό Θόλωμα, αλλαγή σε κλίμακα του γκρι και διαστολή (a) Κατάσταση χωρίς ανοικτά μπλοκ (b) Κατάσταση με διαφορετικά νούμερα και ανοικτά μπλοκ.

Με την ολοκλήρωση της προεπεξεργασίας, καλείται η συνάρτηση ανίχνευσης ακμών με είσοδο δύο συνιστώσες τύπου CV_16SC3 και εύρος 300 έως 455. Με συνάρτηση κατωφλίωσης εύρους για το λευκό χρώμα των ισχυρών ακμών και εύρεσης περιγραμμάτων μπορεί να εντοπιστεί μία συστοιχία εικονοστοιχείων που είναι παρούσα μόνο στα κλειστά μπλοκ [Εικόνα 3.13]. Ομοίως με την εύρεση αριθμών, γίνεται προσέγγιση του περιγράμματος και εύρεση του δείκτη που αντιστοιχεί στο μπλοκ που το περιέχει.



Εικόνα 3.13: Ανίχνευση ακμών Canny στην Εικόνα 3.12 (a) Κατάσταση χωρίς ανοικτά μπλοκ (b) Κατάσταση με διαφορετικά νούμερα και ανοικτά μπλοκ.

3.5.3 Ανάθεση Εξαγόμενων Πληροφοριών

Μετά από την εύρεση τόσο των κλειστών μπλοκ όσο και των αριθμών από τον Αυτοματοποιημένο Επιλυτή, όσα μπλοκ δεν ανήκουν σε καμία από τις δύο κατηγορίες θεωρούνται άμεσα ανοικτά μπλοκ. Στην περίπτωση των αριθμών, αφού βρεθεί ο δείκτης ελέγχεται αν υπάρχει καταχώρηση στη λίστα numbers (με τα ορθογώνια περιγράμματα) για το μπλοκ που επρόκειτο να σημανθεί ως νούμερο. Αν υπάρχει και ο εν λόγω δείκτης είναι εντός ορίων στη λίστα που περιέχει όλα τα μπλοκ (για αποφυγή εξαιρέσεων), ανατίθεται ως κατάσταση και αριθμός ναρκών το νούμερο που περιέχει το μπλοκ. Στην περίπτωση των κλειστών μπλοκ, αρκεί ο δείκτης να είναι εντός ορίων για να προχωρήσει η ανάθεση.

3.6 Βασικές Στρατηγικές

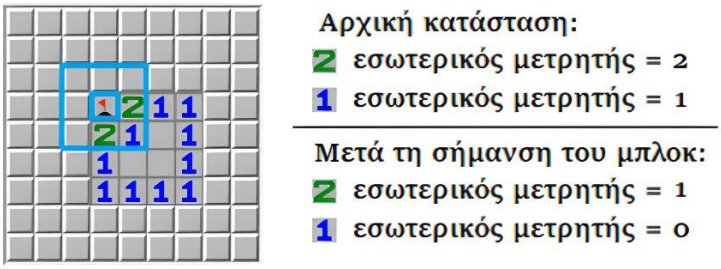
Ξεκινώντας μία παρτίδα, η πρώτη κίνηση είναι πάντα ασφαλής. Οι περισσότεροι παίκτες ενστικτωδώς θα κάνουν την πρώτη τους κίνηση στο κέντρο του ταμπλό όπου θα έχουν και τις περισσότερες πιθανότητες να αποκαλύψουν μεγαλύτερο μέρος του. Παρόλο που συνήθως αυτή η κίνηση προσδίδει περισσότερες πληροφορίες, δε μεταφράζονται απαραίτητα σε μεγαλύτερο ποσοστό επιτυχίας. Όπως αποδεικνύεται από τις εκτενείς προσομοιώσεις της Tu [26], το μεγαλύτερο ποσοστό επιτυχίας βρίσκει η πρώτη κίνηση στις γωνίες του ταμπλό. Στον πίνακα 3.3 αναλύονται τα ευρήματα της για το άνω όριο ποσοστού επιτυχίας για όλες τις βασικές κλίμακες δυσκολίας με βάση την αρχική κίνηση σε τρία διαφορετικά σημεία. Η σύγκριση των κεντρικών σημείων με τις γωνίες κάνει εμφανή τη διαφορά τεσσάρων ποσοστιαίων μονάδων σε ποσοστό επιτυχίας υπέρ των γωνιών.

Δυσκολία	**Γωνία**	**Άκρη**	**Κέντρο**
Beginner	0.9396	0.9117	0.8983
Intermediate	0.9381	0.9108	0.8963
Expert	0.8970	0.8614	0.8530

Πίνακας 3.3: Άνω όρια ποσοστού επιτυχίας για διαφορετικές κλίμακες δυσκολίας και σημεία αρχικής κίνησης [72].

Τα στατιστικά στοιχεία του παραπάνω πίνακα αξιοποιούνται στον Αυτοματοποιημένο επιλυτή, ο οποίος ως αρχική στρατηγική επιλέγει την πάνω αριστερή γωνία. Η αρχή αυτή περιορίζει σημαντικά την πιθανότητα να υπάρχει νάρκη κατά τη δεύτερη κίνηση.

Οι πρώτοι επιλυτές για το Ναρκαλιευτή, όπως αυτός του Adamatzky [15], έκαναν αποκλειστικά χρήση ντετερμινιστικών αλγορίθμων για την εύρεση των μετέπειτα ασφαλών κινήσεων. Ο Pedersen [13] επέκτεινε αυτές τις μεθόδους εισάγοντας τυχαία επιλογή όταν το σύνολο διαθέσιμων κινήσεων είναι κενό. Η υλοποίηση εξετάζει ένα μπλοκ τη φορά και εξάγει συμπεράσματα με βάση την κατάσταση των γειτονικών του μπλοκ. Εφαρμόζονται δύο κύριες στρατηγικές με χρονική πολυπλοκότητα $O((n \times m)^2)$ για ταμπλό $n \times m$ διαστάσεων, η σήμανση ξεκάθαρων ναρκών και η αποκάλυψη απόλυτα ασφαλών μπλοκ, οι οποίες θα αναλυθούν στη συνέχεια. Σημαντική πτυχή και για τις δύο αποτελεί μία από τις ιδιότητες των στοιχείων στη λίστα με τα μπλοκς, οι εναπομένουσες νάρκες. Όπως παρουσιάζεται στην Εικόνα 3.14, κάθε αριθμός κρύβει έναν εσωτερικό μετρητή που εκφράζει τον αριθμό ναρκών που ψάχνει. Με κάθε σήμανση νάρκης, οι μετρητές των μπλοκ που την περιβάλλουν μειώνονται κατά ένα.



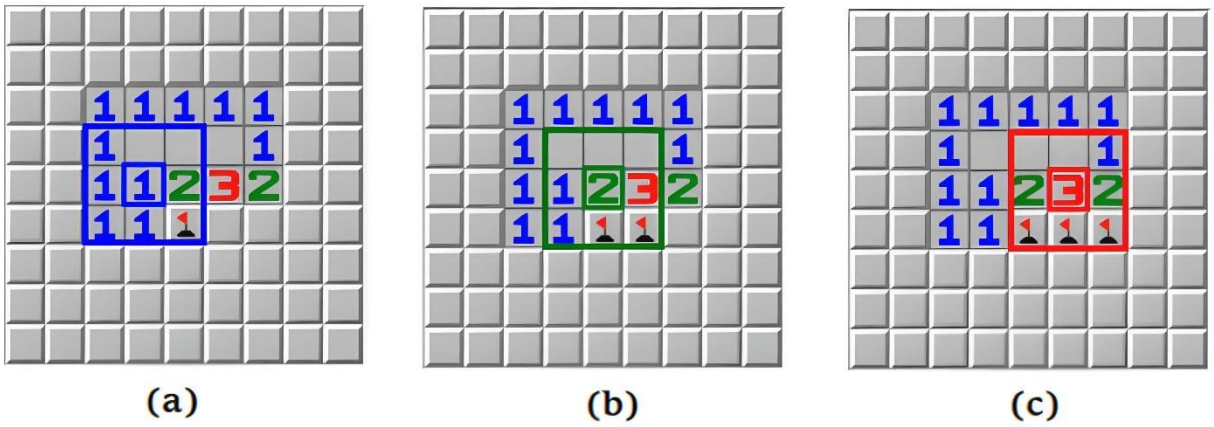
Εικόνα 3.14: Αναπαράσταση μεταβολής εσωτερικού μετρητή ναρκών για μπλοκ αριθμούς.

3.6.1 Σήμανση Ξεκάθαρων Ναρκών

Επιλέγοντας ένα μπλοκ που περιέχει αριθμό, μπορούν να αντληθούν πληροφορίες σχετικά με την κατάσταση που κρύβουν τα γειτονικά κλειστά μπλοκ. Κατά την στρατηγική σήμανσης ξεκάθαρων ναρκών, ελέγχεται διαδοχικά για κάθε μπλοκ που είναι αριθμός και δεν έχει μηδενικό εσωτερικό μετρητή αν ο αριθμός κλειστών γειτονικών μπλοκ είναι ίσος με τις νάρκες που ψάχνει [Αλγόριθμος 3.3]. Αν ισχύει αυτή η συνθήκη, τότε κάθε γειτονικό μπλοκ είναι νάρκη και πρέπει να σημανθεί. Για τη σήμανση, πρέπει να βρεθεί ο δείκτης του εξεταζόμενου μπλοκ και να μειωθούν οι μετρητές των γειτονικών μπλοκ καθώς και ο καθολικός μετρητής ναρκών. Στην Εικόνα 3.15, εξετάζονται τρία μπλοκ με αριθμούς 1,2 και 3. Αρχικά, ο αριθμός 1 γειτονεύει μοναχά με ένα κλειστό μπλοκ οπότε αυτό είναι ξεκάθαρη νάρκη. Γύρω από τη σημαία, μειώνονται οι εσωτερικοί μετρητές κατά ένα. Αυτό σημαίνει πως ο αριθμός 2 πλέον ψάχνει για μία και μόνο νάρκη και αφού γειτονεύει με μία σημαία και ένα κλειστό μπλοκ, το κλειστό μπλοκ είναι νάρκη. Ομοίως, ο αριθμός 3 ψάχνει μία και μόνο νάρκη οπότε σημάνει το τελευταίο κλειστό γειτονικό του μπλοκ ως νάρκη.

```
Algorithm 3: Find Obvious Mines
Data: array blocks: lists all blocks
for i=0; i < blocks.length; i++; do-
    if blocks[i].isNumber then-
        neighbours ← getclosedneighbours(blocks[i]);
        if neighbours.Count(state==closed)==blocks[i].mines then –
            for j=0; j < neighbours.length; j++; do-
                index ← findindex(contours[i].Approximation());
                updatemines(index);
                setinfo(index);
                totalmines – –;
return board;
```

Αλγόριθμος 3.3: Διαδικασία Εύρεσης Ξεκάθαρων Ναρκών – Για κάθε μπλοκ που είναι νούμερο ελέγχεται αν ο αριθμός των κλειστών γειτονικών μπλοκ είναι ίσος με τις νάρκες που αναζητεί. Αν αυτό ισχύει είναι όλες νάρκες.



Εικόνα 3.15: Σήμανση Ξεκάθαρων Ναρκών για τρία μπλοκ-αριθμούς (1,2,3) – (a) Το 1 γειτονεύει με ένα και μόνο κλειστό μπλοκ άρα αυτό είναι η νάρκη (b) Το 2 γειτονεύει με μια σημαία και ένα κλειστό μπλοκ και ο μετρητής του έχει μειωθεί κατά 1, άρα το κλειστό μπλοκ είναι νάρκη (c) Ομοίως για το 3, ο μετρητής του έχει μειωθεί κατά 2 άρα το κλειστό μπλοκ που γειτονεύει είναι νάρκη.

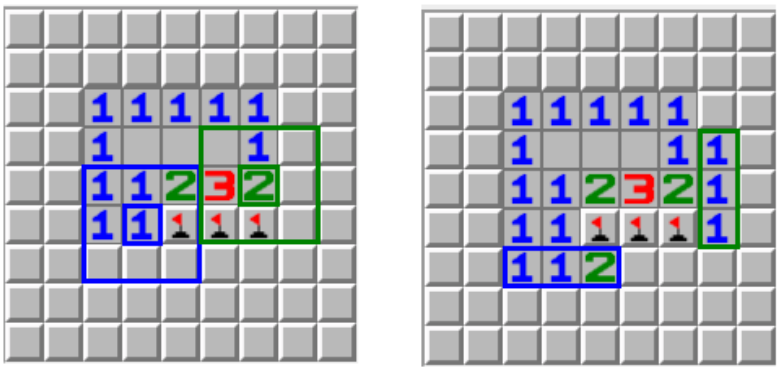
3.6.2 Αποκάλυψη Απόλυτα Ασφαλών Μπλοκ

Όπως είναι προφανές, μετά από την εκτέλεση της πρώτης στρατηγικής κάποια μπλοκ με αριθμούς έχουν μηδενίσει τους εσωτερικούς μετρητές και γειτονεύουν μόνο με νάρκες οπότε δε χρειάζεται να ληφθούν υπόψη στην αναζήτηση για επόμενες κινήσεις. Υπάρχουν, όμως, ορισμένοι αριθμοί που έχουν συμπληρώσει τις νάρκες που έψαχναν αλλά εξακολουθούν να γειτονεύουν με κλειστά μπλοκ. Σε αυτό το σημείο ξεκινάει η δράση της δεύτερης στρατηγικής. Ελέγχει τη λίστα με το σύνολο των μπλοκ και βρίσκει ποια από αυτά είναι νούμερα που γειτονεύουν με αριθμό σημαιών ίσο με τον νούμερο που εκφράζουν [Αλγόριθμος 3.4]. Στη συνέχεια, αποκαλύπτονται όλα τα κλειστά μπλοκ γύρω από όσα ικανοποιούν αυτές τις συνθήκες. Η Εικόνα 3.16 αποτελεί τη συνέχεια της εκτέλεσης αφού

γίνει η σήμανση των ξεκάθαρων ναρκών. Έχει γίνει επισήμανση των αριθμών που προκαλούν αποκαλύψεις και το πεδίο επιρροής τους.

```
Algorithm 4: Probe Safe Blocks
Data: array blocks: lists all blocks
for i=0; i < blocks.length; i++; do-
    if blocks[i].isNumber then-
        neighbours ← getclosedneighbours(blocks[i]);
        flags ← getflaggedneighbours(neighbours);
        if flags.Count==blocks[i].state then –
            for j=0; j < flagged.length; j++; do-
                index ← findindex(contours[i].Approximation());
                click(index);
```

Αλγόριθμος 3.4: Διαδικασία Αποκάλυψης Ασφαλών Μπλοκ – Για κάθε μπλοκ που είναι νούμερο ελέγχεται αν ο αριθμός των γειτονικών σημαιών είναι ίσος με τον αριθμό που εκφράζει. Αν αυτό ισχύει είναι όλα ασφαλή.



Εικόνα 3.16: Αποκάλυψη απόλυτα ασφαλών μπλοκ – (a) Έχουν βρεθεί και επισημανθεί δυο αριθμοί που έχουν συμπληρώσει τον αριθμό ναρκών τους (b) Τα κλειστά μπλοκ που αποκαλύφθηκαν ήταν απόλυτα ασφαλή.

3.7 Πρόβλημα Ικανοποίησης Περιορισμών

Το κύριο μειονέκτημα κάθε υλοποίησης μόνο με βασικές στρατηγικές είναι πως σε ορισμένες διαμορφώσεις του ταμπλό δεν επαρκούν για να αποφασιστεί η επόμενη κίνηση. Αν ο επιλυτής είναι εξ ολοκλήρου ντετερμινιστικός, δε λαμβάνει τυχαίες αποφάσεις (ή έστω εμπεριστατωμένες εικασίες) κάτι το οποίο είναι σχεδόν αναπόφευκτο σε υψηλότερες δυσκολίες. Σε αυτό το σημείο αναλαμβάνει ένας αλγόριθμος μοντελοποίησης και επίλυσης ως CSP. Ένα τέτοιο μοντέλο παρουσίασε ο Studholme [24], υπολογίζοντας τον αριθμό συνεπών και πλήρων αναθέσεων για το εκάστοτε μπλοκ. Αυτό επέτρεπε τον υπολογισμό της πιθανότητας να υπάρχει νάρκη στο υπό εξέταση μπλοκ. Το κύριο πρόβλημα της υλοποίησης του Studholme είναι πως ο χώρος αναζήτησης πιθανών να είναι μεγάλος. Την αδυναμία αυτή

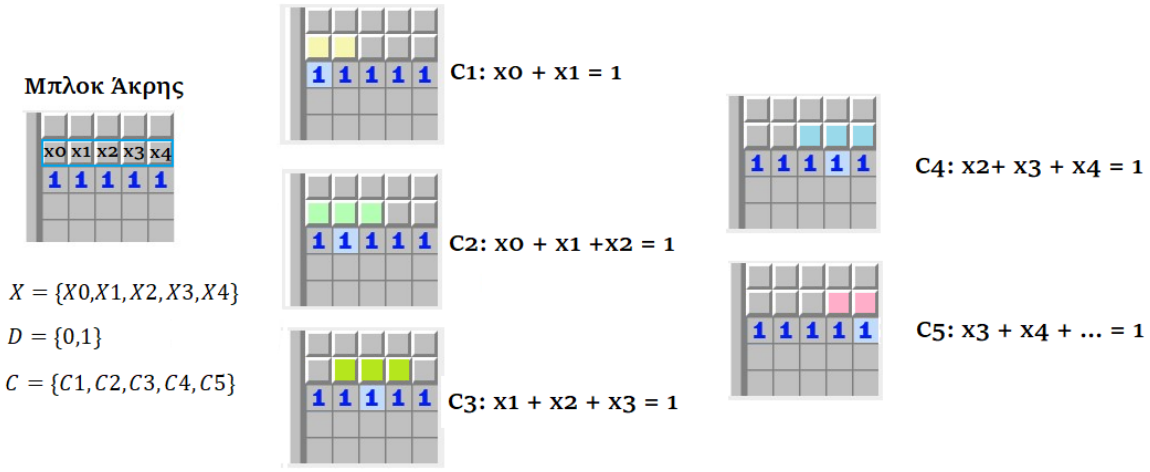
καλύπτει ο επιλυτής CP-SAT σε συνδυασμό με τη γεννήτρια πιθανοτήτων του Αυτοματοποιημένου επιλυτή. Η επιλογή με βάση την πιθανότητα να υπάρχει νάρκη για κάθε μπλοκ εκμεταλλεύεται πλήρως την κατάσταση του ταμπλό και σίγουρα επιφέρει καλύτερα αποτελέσματα από την τυχαία επιλογή.

3.7.1 Μοντελοποίηση του Προβλήματος

Όταν οι βασικές στρατηγικές φτάσουν σε μια κατάσταση του ταμπλό που δε μπορούν να επιλύσουν, καλείται η συνάρτηση μοντελοποίησης σε CSP με χρονική πολυπλοκότητα $O((n \times m)^2)$ για ταμπλό n×m διαστάσεων. Με είσοδο την κατάσταση του ταμπλό, ξεκινά η έκφραση της ως τα τρία βασικά κομμάτια $\langle X, D, C \rangle$ του CSP. Το μοντέλο περιλαμβάνει ως μεταβλητές X τα μπλοκ άκρης [Εικόνα 3.17] τα οποία είναι όσα κλειστά μπλοκ γειτονεύουν με κάποιο αριθμό. Εισάγοντας μόνο τα μπλοκ άκρης περιορίζεται η αχρείαστη επέκταση του χώρου αναζήτησης και κατά συνέπεια ο χρόνος εκτέλεσης. Το πεδίο ορισμού D περιλαμβάνει τις τιμές μηδέν και ένα εκφράζοντας την ύπαρξη (ή μη) της νάρκης. Το σύνολο των αριθμών επιβάλλουν περιορισμούς C στα μπλοκ άκρης με τη μορφή:

$$C_i \rightarrow X_1 + X_2 + \ldots X_8 = Μετρητής\ Ναρκών \qquad \textbf{(3.8)}$$

Σύμφωνα με την εξίσωση 3.8, κάθε περιορισμός περιέχει έως 8 μεταβλητές (επειδή μπορεί να έχει έως 8 γείτονες) και είναι γραμμική εξίσωση με βάση τις νάρκες που αναζητεί ο αριθμός. Στην Εικόνα 3.17 περιγράφεται μία κατάσταση του ταμπλό με πέντε μεταβλητές X_{0-4}, πεδία ορισμού $D_{0-4} = \{0,1\}$ και πέντε αριθμούς που εκφράζονται ως περιορισμοί C_{1-5} με τη γραμμική εξίσωση των μεταβλητών X_{0-4} και τον αντίστοιχο μετρητή.



Εικόνα 3.17: Μοντελοποίηση μιας κατάστασης του Ναρκαλιευτή ως CSP. Σαν μεταβλητές ορίζονται τα μπλοκ άκρης, ως πεδίο ορισμού το 0 και 1 και σαν περιορισμοί οι γραμμικές εξισώσεις που επιβάλλουν οι αριθμοί.

3.7.2 Εύρεση Λύσεων με CP-SAT

Αμέσως μετά τη μοντελοποίηση του προβλήματος, το CP-SAT ενεργοποιείται για την εύρεση των συνεπών και πλήρων αναθέσεων για τα μπλοκ άκρης. Το μοντέλο κωδικοποιείται (με χρονική πολυπλοκότητα $O(n^2)$) από το LCG για να μπορεί να επιλυθεί ως SAT ($O(2^n)$) ενώ παράλληλα να μπορεί να γίνει προεπεξεργασία του. Με αυτό τον τρόπο μειώνονται τα πεδία ορισμού και απλοποιούνται οι περιορισμοί που θα κληθούν να επιβάλλουν οι μηχανισμοί διάδοσης κατά την επίλυση. Με την προεπεξεργασία, ομαδοποιούνται εμμέσως οι περιορισμοί που μοιράζονται μεταβλητές. Με τη χρήση του μηχανισμού VSIDS, εξερευνούνται πρώτα τα κυριολεκτικά με τις περισσότερες εμφανίσεις στις πρόσφατα εξεταζόμενες προτάσεις. Όπως στην περίπτωση του X_1 που εμφανίζεται σε τρείς προτάσεις/περιορισμούς στο παράδειγμα της Εικόνας 3.17, οι αναθέσεις κυριολεκτικών με μεγάλη βαθμολογία συνήθως έχουν σημαντικό αντίκτυπο οπότε πρέπει να εξεταστούν νωρίς για αποφυγή συγκρούσεων.

Το CP-SAT δημιουργεί το εσωτερικό μοντέλο για τον επιλυτή SAT ενώ ταυτόχρονα εξάγει τις επιλογές του VSIDS στο μοντέλο διοχέτευσης για να συμπληρωθεί κάθε πιθανός συνδυασμός αναθέσεων. Με την εύρεση αναθέσεων που ικανοποιούν όλους τους περιορισμούς σχηματίζονται οι πλήρεις και συνεπείς λύσεις οι οποίες αξιοποιούν ένα συγκεκριμένο αριθμό ναρκών [Εικόνα 3.18]. Οι αναθέσεις μίας λύσης αφορούν αποκλειστικά τα μπλοκ άκρης. Για κάθε λύση μπορούν να υπολογιστούν όλοι οι πιθανοί συνδυασμοί του ταμπλό που περιλαμβάνουν τις αναθέσεις της όπως περιγράφει η εξίσωση 3.9.

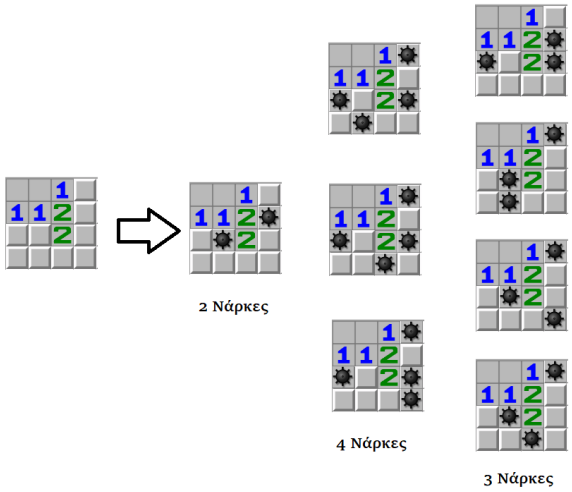
$$Συνδυασμοί\ C(n,r) = \frac{n!}{(r!(n-r)!)}, \begin{cases} n = κλειστά\ μπλοκς - μπλοκ\ άκρης \geq 0 \\ r = καθολικές\ νάρκες - νάρκες\ λύσης \geq 0 \end{cases}, n > r \quad \textbf{(3.9)}$$

Για λόγους σαφήνειας αυτοί θα αποκαλούνται «τοπικοί συνδυασμοί». Καθώς οι υπολογισμοί μπορεί να αφορούν αριθμούς με εκατοντάδες ψηφία είναι δαπανηροί από άποψη υπολογιστικών πόρων. Για παράδειγμα, οι πιθανοί συνδυασμοί αναθέσεων στην αρχή της παρτίδας για την κλίμακα δυσκολίας «Expert» είναι $C(480,99) = 5.6 \times 10^{104}$. Το κάθε ζεύγος n και r αποθηκεύεται σε μία λίστα μαζί με το αποτέλεσμά του ώστε να μπορεί να επαναχρησιμοποιηθεί και να εξοικονομηθούν πόροι. Οι τοπικοί συνδυασμοί προστίθενται σε κάθε μπλοκ στο οποίο είχε ανατεθεί νάρκη κατά την λύση. Παράλληλα, προστίθενται στους «καθολικούς συνδυασμούς» οι οποίοι συμπεριλαμβάνουν τους τοπικούς συνδυασμούς από κάθε πιθανή λύση που επέστρεψε το CP-SAT. Η πιθανότητα ένα μπλοκ άκρης να περιέχει νάρκη υπολογίζεται από την παρακάτω εξίσωση 3.10:

$$P(\text{να είναι νάρκη σε μπλοκ άκρης}) = \frac{\text{Τοπικοί Συνδυασμοί}}{\text{Καθολικοί Συνδυασμοί}} \qquad \textbf{(3.10)}$$

Για τον υπολογισμό της ανάλογης πιθανότητας για τα μπλοκ που είναι κλειστά αλλά δεν ανήκουν στα μπλοκ άκρης αρκεί να εφαρμοστεί ο τύπος της πυκνότητας ναρκών [Εξίσωση 3.11] συναρτήσει των νέων δεδομένων ανάλογα με την κατάσταση της παρτίδας.

$$P(\text{να είναι νάρκη εκτός άκρης}) = \frac{\text{Καθολικός Μετρητής Ναρκών}}{\text{Κλειστά Μπλοκ}} \qquad \textbf{(3.11)}$$



Εικόνα 3.18: Οι πιθανές αναθέσεις ναρκών που σχηματίζουν πλήρεις λύσεις για το CSP πάνω στα μπλοκ άκρης. Παρατηρείται χρήση διαφορετικού αριθμού ναρκών άρα και τοπικών συνδυασμών.

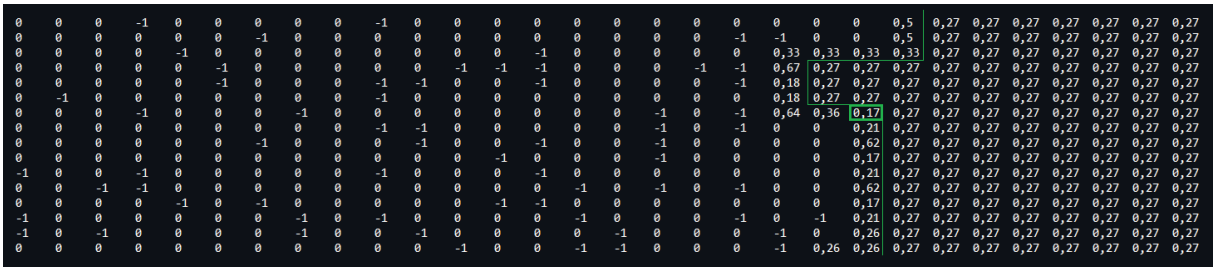
3.7.3 Σήμανση Στατιστικά Σίγουρων Κινήσεων

Έχοντας ολοκληρώσει τον υπολογισμό των πιθανοτήτων για κάθε ένα από τα κλειστά μπλοκ, ο Αυτοματοποιημένος Επιλυτής είναι σε θέση να εκτελέσει κινήσεις με στατιστική σιγουριά. Σε περίπτωση που κάποιο μπλοκ ανατίθεται σε κάθε λύση ως νάρκη, σημαίνει πως οι τοπικοί του συνδυασμοί είναι ίσοι με τους καθολικούς. Κατά επέκταση, αυτό μεταφράζεται σε 100% πιθανότητα να είναι νάρκη οπότε προστίθεται άμεσα σε λίστα για να σημανθεί ως νάρκη. Στην περίπτωση που κάποιο μπλοκ δεν ανατίθεται σε καμία λύση ως νάρκη, έχει 0% πιθανότητα και είναι απόλυτα ασφαλές να αποκαλυφθεί.

3.7.4 «Τυχαίες» Κινήσεις

Όταν δεν είναι διαθέσιμες στατιστικά σίγουρες κινήσεις ο επιλυτής πρέπει να καταφύγει σε εμπεριστατωμένες εικασίες (educated guesses). Οι επιλογές αυτές δεν είναι τυχαίες αλλά γίνονται με βάση τις πιθανότητες. Ο επιλυτής ελέγχει τις διαθέσιμες κινήσεις και επιλέγει προς αποκάλυψη το μπλοκ με τις χαμηλότερες πιθανότητες να είναι νάρκη. Σε περίπτωση ισοψηφίας επιλέγεται το μπλοκ που είναι πιο κοντά στην άνω αριστερή γωνία.

Αυτή είναι η μόνη στρατηγική του επιλυτή που ενέχει ρίσκο καθώς όλες οι υπόλοιπες κινούνται στα πλαίσια των ντετερμινιστικών αλγορίθμων. Η εσωτερική αποτύπωση του πίνακα πιθανοτήτων για το ταμπλό εμφανίζεται στην Εικόνα 3.19 για μία τυχαία κατάσταση.

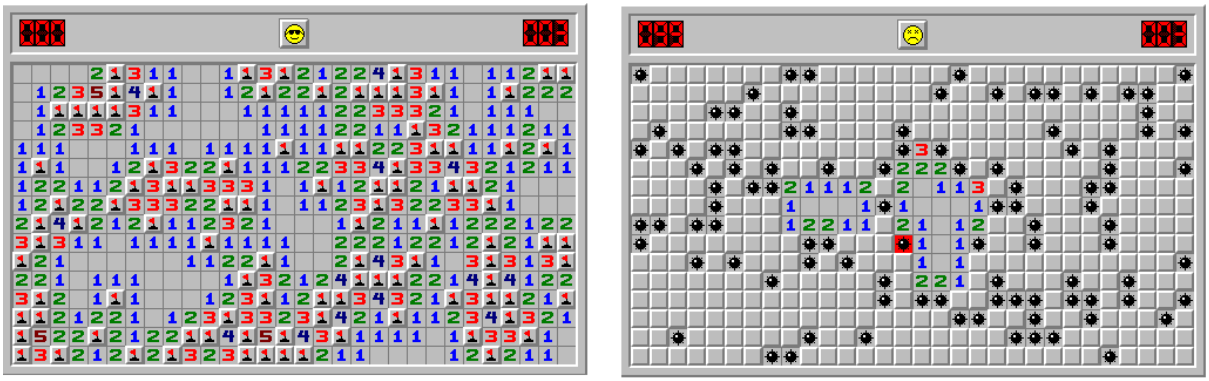


Εικόνα 3.19: Η εσωτερική αποτύπωση του ταμπλό για τις πιθανότητες του κάθε μπλοκ να είναι νάρκη. Ως -1 εκφράζονται οι σημαίες, ως 0 τα ανοιχτά μπλοκ και ως 0.27 τα κλειστά μπλοκ εκτός άκρης. Τα μπλοκ άκρης διαχωρίζονται με μία γραμμή ενώ η πιο ασφαλής κίνηση έχει επισημανθεί (0.17).

3.8 Συνθήκες Τέλους Παιχνιδιού

Μετά την εφαρμογή τόσο των βασικών στρατηγικών όσο και της προσέγγισης ως CSP, είναι απαραίτητο να ελεγχθεί αν πληρούνται οι συνθήκες τερματισμού παρτίδας. Αν ο συνδυασμός των στρατηγικών εφαρμοστεί επιτυχώς έως το τέλος της παρτίδας, ο επιλυτής έχει κερδίσει και παρουσιάζεται η κατάσταση της Εικόνας 3.20a. Από την άλλη, αν ο επιλυτής επιλέξει νάρκη στην κίνηση που επέφερε κάποια πιθανοτική ανάλυση, χάνει την παρτίδα και συναντάει την κατάσταση της Εικόνας 3.20b. Δεδομένου ότι ο Αυτοματοποιημένος επιλυτής ακολουθεί τακτική No-Flag (δεν αναθέτει σημαίες καθώς τις αποτυπώνει εσωτερικά), οι σημαίες που εμφανίζονται όταν αποκαλυφθούν όλα τα κλειστά μπλοκ χωρίς νάρκη μπορούν να αξιοποιηθούν για την αναγνώριση της νίκης. Παράλληλα, το κόκκινο πλαίσιο γύρω από την νάρκη που ευθύνεται για τον πρόωρο τερματισμό της παρτίδας μπορεί να χρησιμοποιηθεί για την αναγνώριση της ήττας. Καθώς και τα δύο αντικείμενα που πρέπει να ανιχνευθούν έχουν σχεδόν το ίδιο χρώμα, η ειδοποιός τους διαφορά βρίσκεται στο εμβαδόν τους. Ωστόσο, πρέπει να ληφθεί υπόψη και ο αριθμός 3 διότι το εύρος τιμών έντασης του καλύπτει το εύρος των συνθηκών τερματισμού.

Εφαρμόζοντας αλγόριθμο κατωφλίωσης με εύρος (0,0,160)-(60,60,255), όπως για τον αριθμό 3, μπορεί να εκτελεστεί αλγόριθμος για την αποτελεσματική εύρεση περιγραμμάτων των αντικειμένων υπό εξέταση. Στη συνέχεια, ελέγχονται όλα τα περιγράμματα και αντιστοιχίζονται με βάση το εμβαδόν. Συγκεκριμένα, οι σημαίες φιλτράρονται με εμβαδόν μεταξύ 2 και 8 εικονοστοιχείων ενώ η αποκαλυμμένη νάρκη με εμβαδόν μεγαλύτερο των 40 εικονοστοιχείων. Κατά αυτόν τον τρόπο, αποφεύγεται η λανθασμένη αναγνώριση του αριθμού 3 ως συνθήκη τερματισμού ενώ διαχωρίζεται η νίκη με την ήττα.



(a) (b)

Εικόνα 3.20: Οι καταστάσεις τερματισμού παρτίδας (a) Τερματισμός παρτίδας ως νίκη (b) Τερματισμός παρτίδας ως ήττα.

3.9 Σύνοψη κεφαλαίου

Στο κεφάλαιο αυτό έγινε παρουσίαση της διεπαφής του χρήστη και αναφέρθηκαν οι επιμέρους μέθοδοι ως συνδυασμοί αλγορίθμων για την επίλυση των υποπροβλημάτων που προκύπτουν κατά την επίλυση μίας παρτίδας Ναρκαλιευτή. Παράλληλα, παρατέθηκε η σειρά εκτέλεσης των μεθόδων καθώς και η ροή δεδομένων μεταξύ σταδίων της επίλυσης. Στο επόμενο κεφάλαιο παρουσιάζεται η πειραματική διαδικασία για τον έλεγχο ορθότητας λειτουργίας, την εξαγωγή στατιστικών στοιχείων και τη διαμόρφωση παρατηρήσεων.

Κεφάλαιο 4

Πειραματική Διαδικασία

Στο κεφάλαιο αυτό παρουσιάζεται η διαδικασία για την εξαγωγή στατιστικών στοιχείων και μετρικών από ένα σύνολο παρτίδων. Εξετάζονται οι προσαρμογές στην προγραμματιστική υλοποίηση, οι μετρήσεις κατά τη διάρκεια μίας παρτίδας και τα συμπεράσματα που διαμορφώνονται με τη μελέτη των αποτελεσμάτων.

4.1 Ορθή λειτουργία και Περιβάλλον πειράματος

Κατά τον προγραμματισμό του Αυτοματοποιημένου επιλυτή, έγινε έλεγχος ορθής λειτουργίας σε τέσσερεις υπολογιστές με διαφορετικές τεχνικές προδιαγραφές. Ως βασικοί κανόνες ελέγχου ήταν η ανάλυση της οθόνης 1920×1080 σε 100% κλίμακα και η απουσία χρωματικών φίλτρων. Διαδοχικά, έγινε σε κάθε συσκευή ο έλεγχος για τον εντοπισμό του ταμπλό και την αναγνώριση της κατάστασης των μπλοκ. Τα εύρη έντασης για κάθε αριθμό προσαρμόστηκαν με βάση τα ευρήματα ώστε να εγγυάται η ακρίβεια σε ίδιο βαθμό για κάθε συσκευή που δοκιμάστηκε. Σε επόμενες φάσεις του ελέγχου, δόθηκε σε κάθε συσκευή το ίδιο πρόβλημα με τη μορφή ορισμένου ταμπλό για να ελεγχθεί πως εξάγουν την ίδια λύση.

Επεξεργαστής	Μνήμη RAM	Κάρτα Γραφικών	Μέγεθος Οθόνης	Έκδοση Windows
AMD Ryzen 7 3750H 2.3GHz	16GB	NVIDIA GeForce GTX 1650	17”	W11 64-bit
AMD Ryzen 5 2600 3.4GHz	16GB	NVIDIA GeForce GTX 1050 Ti	27”	W10 64-bit
Intel Core i7-8750H 2.2GHz	16GB	NVIDIA GeForce GTX 1060 Max-Q	17”	W11 64-bit
Intel Core i7-11700K 3.6GHz	16GB	NVIDIA GeForce RTX 2080 OC	27”	W11 64-bit

Πίνακας 3.4: Τεχνικές Προδιαγραφές υπολογιστών που έγιναν δοκιμές ορθής λειτουργίας.

Ο επιλυτής τροποποιήθηκε ώστε οι λειτουργίες του να παρέχονται δίχως γραφικό περιβάλλον και μέσω της γραμμής εντολών των Windows. Με αυτή την υλοποίηση εξοικονομείται ο χρόνος που απαιτείται για την προβολή και την αλληλεπίδραση με το μενού.

Ως μοναδικός περιορισμός για την εκτέλεση τέθηκε ο μέγιστος χρόνος των 120 δευτερολέπτων ανά παρτίδα. Σε αυτές τις περιπτώσεις το ταμπλό επαναφέρεται στην αρχική του κατάσταση και δεν καταγράφονται στατιστικά ανεξάρτητα από το πιθανό αποτέλεσμα. Για τη διεκπεραίωση της πειραματικής διαδικασίας τηρήθηκαν οι προϋποθέσεις ανάλυσης και κλίμακας ενώ έγινε χρήση υπολογιστή με τα εξής τεχνικά χαρακτηριστικά: Επεξεργαστής AMD Ryzen 7 3750H 2.3GHz , 16GB μνήμη RAM, Κάρτα γραφικών NVIDIA GeForce GTX 1650, Οθόνη 17 ιντσών Windows 11 64-bit.

4.2 Συλλογή Στατιστικών Στοιχείων

Για τη συλλογή στατιστικών στοιχείων, ολοκληρώθηκαν 6500 παρτίδες ναρκαλιευτή από τον Αυτοματοποιημένο Επιλυτή. Οι ιδιότητες που καταγράφηκαν σε κάθε παρτίδα ήταν: (α) το αποτέλεσμα (Νίκη/Ηττα), (β) το σύνολο των κλικ, (γ) οι ενεργοποιήσεις του CP-SAT, (δ) οι «τυχαίες» κινήσεις, (ε) οι νάρκες που απέμειναν, (ζ) ο χρόνος ολοκλήρωσης (CPU Time). Σε ξεχωριστές δοκιμές έγινε συλλογή στοιχείων σχετικά με το χρόνο που απαιτεί η εύρεση και αναγνώριση του ταμπλό. Τα στοιχεία αυτά συλλέχθηκαν κυρίως από τα ταμπλό με διαστάσεις 9x9, 16x16 και 30x16 αλλά και από παραλλαγές τους που προέκυψαν με την εφαρμογή πυκνότητας που αντιστοιχεί σε κάθε δυσκολία. Για παράδειγμα, στο ταμπλό της κλίμακας «Beginner» εφαρμόζεται η πυκνότητα του «Expert» με τα ανάλογα αποτελέσματα.

4.3 Χρόνος Εύρεσης και Αναγνώρισης του ταμπλό

Η κύρια κατάκτηση του Αυτοματοποιημένου Επιλυτή έρχεται άμεσα σε σύγκριση με την υλοποίηση του Packard [29]. Σύμφωνα με τον ίδιο, οι συναρτήσεις εντοπισμού με φωτογραφίες για κάθε κατάσταση απαιτούν 13ms. Αυτό μεταφράζεται σε 130ms δεδομένου ότι υπάρχουν δέκα πιθανές καταστάσεις για ένα μπλοκ (1-8 και Ανοικτό/Κλειστό). Τα αποτελέσματα της μεθόδου που εφαρμόζει η παρούσα εργασία παρουσιάζονται στον Πίνακα 4.1 μαζί με τις μετρήσεις για την εύρεση του ταμπλό (που δεν υλοποιείται από τον Packard).

	Χρόνος/Ταμπλό	**Εύρεση Ταμπλό**
9x9, 81 μπλοκς	16.01 ms	116.82 ms
16x16, 256 μπλοκς	17.99 ms	121.41 ms
30x16, 480 μπλοκς	24.17 ms	131.56 ms

Πίνακας 4.1: Χρόνοι εύρεσης και αναγνώρισης κατάστασης του ταμπλό για τα μεγέθη των βασικών κλιμάκων.

Αξίζει να σημειωθεί πως ο χρόνος που αναγράφεται στη δεύτερη στήλη αντιστοιχεί στην αναγνώριση και των δέκα καταστάσεων. Ο χρόνος αυτός είναι σημαντικά μειωμένος συγκριτικά με τη μέθοδο των φωτογραφιών αλλά παράλληλα παρουσιάζει εξάρτηση από το σύνολο των μπλοκς και κατά επέκταση την επιφάνεια της φωτογραφίας. Αυτό συμβαίνει, εν μέρει, διότι δεν ελέγχεται όλη η οθόνη, αλλά ένα στιγμιότυπο προσαρμοσμένο ακριβώς στην περιοχή ενδιαφέροντος (Region of Interest). Κατά την εύρεση του ταμπλό, παρατηρείται μία μικρή αύξηση όσο μεγαλώνουν οι διαστάσεις. Η συνάρτηση μετασχηματισμού γραμμών Hough καλείται να ελέγξει γύρω από μεγαλύτερο αριθμό ακμών αυξάνοντας ελαφρώς το χρόνο εκτέλεσής της. Ακόμα και σε αισθητά μεγαλύτερες διαστάσεις δεν υπάρχει προσαύξηση που να δυσχεραίνει την ομαλή εκτέλεση.

4.4 Επιδόσεις στις βασικές κλίμακες δυσκολίας

Η πειραματική διαδικασία επικεντρώθηκε κυρίως στις βασικές κλίμακες δυσκολίας καθώς, όπως θα αποδειχθεί και στην επόμενη ενότητα, οι απαιτητικότερες καταστάσεις παρουσιάζουν αύξηση σε χρονικό κόστος που δε μπορεί να αγνοηθεί. Ο Αυτοματοποιημένος Επιλυτής σημειώνει 90.9% ποσοστό επιτυχίας στην κλίμακα 9x9 «Beginner», 81.1% στο 16x16 «Intermediate» και 44.5% στο 30x16 «Expert». Στην κλίμακα «Expert», που αποτελεί και τη μεγαλύτερη πρόκληση, ο επιλυτής της εργασίας σημειώνει 4.44% μεγαλύτερο ποσοστό επιτυχίας συγκριτικά με τις εργασίες που μελετήθηκαν. Φυσικά, πρέπει να αναφερθεί πως κάθε υλοποίηση διαφέρει ως προς τις μεθόδους και τις χρονικές απαιτήσεις τους αλλά όλες προσφέρουν μια καινούργια ματιά ανεξαρτήτως ποσοστού επιτυχίας. Στον Πίνακα 4.2 παραθέτονται αναλυτικά μέσοι όροι από τις κύριες ιδιότητες κάθε παρτίδας.

Κλίμακα	Σύνολο Κλικ	Χρήσεις CP-SAT	«Τυχαίες» Κινήσεις	Εναπομένουσες Νάρκες	Χρόνος Επίλυσης
9x9	13.661	2.653	0.415	0.765	400ms
16x16	39.431	4.326	1.005	6.432	1071ms
30x16	83.999	17.719	3.532	39.536	4134ms

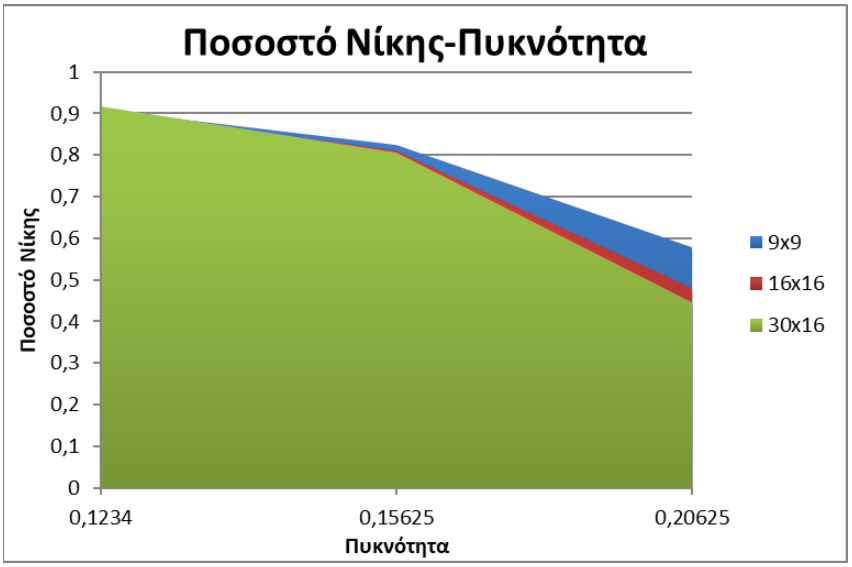
Πίνακας 4.2: Μέσοι όροι από τα κύρια στατιστικά στοιχεία κάθε παρτίδας για τις βασικές κλίμακες δυσκολίας.

4.5 Επιπτώσεις πυκνότητας και διαστάσεων του ταμπλό

Από την εξέταση του παραπάνω πίνακα, είναι εμφανές πως όσο ανεβαίνει η δυσκολία τόσο αυξάνεται η τιμή για κάθε μία από τις ιδιότητες. Αυτό οφείλεται στην αύξηση των μπλοκ αλλά κυρίως στην πυκνότητα των ναρκών που απαρτίζουν το ταμπλό. Η πυκνότητα γίνεται αισθητή κατά την επίλυση λόγω των απαιτητικών καταστάσεων που επιφέρει και ειδικά από τη μείωση της πιθανότητας για ένα κλικ να αποκαλύψει μεγαλύτερη περιοχή. Η αποκάλυψη μικρότερων περιοχών μεταφράζεται ως λιγότερες πληροφορίες για τον επιλυτή άρα και τη μεγαλύτερη ανάγκη για επίλυση ως CSP. Ταυτόχρονα, όσο μεγαλύτερη η πυκνότητα τόσο περισσότερες εμφανίσεις κάνουν αριθμοί (άρα και περισσότεροι περιορισμοί για το μοντέλο CSP) αλλά και ζεύγη μπλοκ που έχουν εξίσου 50% πιθανότητα να περιέχουν νάρκη. Αυτό, για παράδειγμα, σημαίνει πως συνήθως στην πυκνότητα 20.625% (Expert) συσσωρεύεται μεγαλύτερο ρίσκο έως το τέλος της παρτίδας από ότι στην πυκνότητα 15.625% (Intermediate) κάτι το οποίο αντικατοπτρίζεται και στο ποσοστό επιτυχίας. Στον Πίνακα 4.3 και στην Εικόνα 4.1 παρουσιάζονται ταμπλό με διαστάσεις των βασικών κλιμάκων και πυκνότητα από 0.12345 έως 0.20625 (Beginner έως Expert).

	0,12345	**0,15626**	**0.20625**
9x9	0.909	0.824	0.578
16x16	0.91	0.811	0.48
30x16	0.916	0.806	0.445

Πίνακας 4.3: Ποσοστό Νίκης για διαφορετικές τιμές πυκνότητας σε ταμπλό με διαστάσεις των βασικών δυσκολιών.



Εικόνα 4.1: Ποσοστό Νίκης για διαφορετικές τιμές πυκνότητας στα ταμπλό των βασικών δυσκολιών.

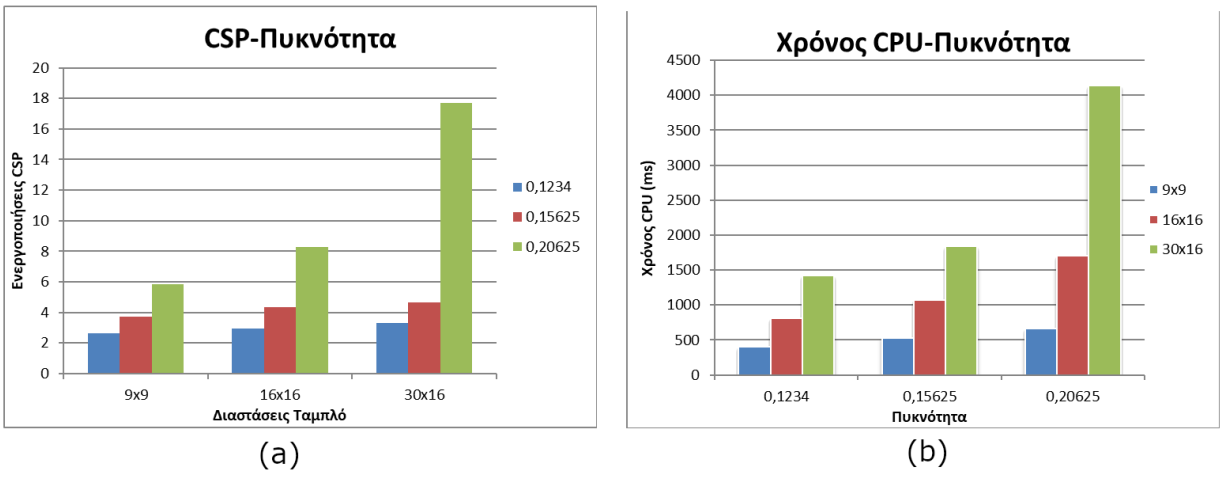
Εξετάζοντας το παραπάνω σχήμα αλλά και τις τιμές του Πίνακα 4.3, επαληθεύεται η πρόταση πως η πυκνότητα έχει αντίκτυπο στο ποσοστό νίκης αλλά ταυτόχρονα παρατηρείται διακύμανση μεταξύ των κλιμάκων του ταμπλό. Μεταξύ των αποτελεσμάτων διαφορετικών διαστάσεων με σταθερή την υψηλότερη πυκνότητα, υπάρχει διαφορά 13.3% κάτι το οποίο δείχνει πως παίζει ρόλο και ο αριθμός των μπλοκ. Σε περισσότερα μπλοκ υπάρχει μεγαλύτερη πιθανότητα να υπάρξουν αναθέσεις ναρκών που οδηγούν σε δυσεπίλυτες καταστάσεις. Οι καταστάσεις αυτές απαιτούν την επίλυση ως CSP και κατά συνέπεια επιπρόσθετο χρόνο CPU. Στους Πίνακες 4.4 και 4.5 καθώς και στην Εικόνα 4.2 αναλύονται οι ενεργοποιήσεις του CP-SAT και ο χρόνος CPU για τις εννέα περιπτώσεις.

	0,12345	**0,15626**	**0.20625**
9x9	2.653	3.722	5.828
16x16	2.964	4.326	8.298
30x16	3.318	4.674	17.719

Πίνακας 4.4: Ενεργοποίηση CP-SAT για διαφορετικές πυκνότητες στις βασικές διαστάσεις ταμπλό.

	0,12345	**0,15626**	**0.20625**
9x9	400.796 ms	529.448 ms	662.048 ms
16x16	806.208 ms	1071.184 ms	1702.168 ms
30x16	1422.142 ms	1836.28 ms	4143.328 ms

Πίνακας 4.5: Χρήση χρόνου CPU για διαφορετικές πυκνότητες στις βασικές διαστάσεις ταμπλό.



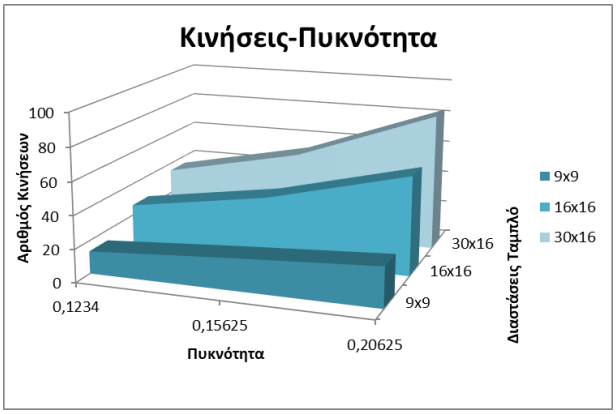
Εικόνα 4.2: Επιπτώσεις της πυκνότητας και των διαστάσεων στο χρόνο CPU και στην ενεργοποίηση του CP-SAT - (a) Επίλυση ως CSP ανά κλίμακα για κάθε πυκνότητα (b) Χρόνος CPU ανά κλίμακα για κάθε πυκνότητα (ms).

Με τη μελέτη των στατιστικών δεδομένων που προσφέρουν οι πίνακες 4.4 και 4.5, αποδεικνύεται η πρόταση πως τόσο η πυκνότητα όσο και το μέγεθος του ταμπλό επηρεάζουν τον αριθμό εμφανίσεων απαιτητικών καταστάσεων. Η διαφορά είναι ιδιαίτερα αισθητή στο ταμπλό 30x16 με την υψηλότερη πυκνότητα καθώς εκεί υπάρχει μεγαλύτερη πιθανότητα να βρεθούν δυσεπίλυτες αναθέσεις ναρκών. Δεδομένου των χρονικών απαιτήσεων της εύρεσης αναθέσεων από το CP-SAT, αξίζει να σημειωθεί πως υπάρχει συσχέτιση μεταξύ της επίλυσης ως CSP και του χρόνου CPU όπως είναι εμφανές και από τα σχήματα της Εικόνας 4.2.

Ένας ακόμα παράγοντας που έχει εξίσου μεγάλη επιρροή το μέγεθος του ταμπλό και η πυκνότητα είναι οι κινήσεις που απαιτούνται για την ολοκλήρωση της παρτίδας [Εικόνα 4.3]. Καθώς ο Αυτοματοποιημένος Επιλυτής ακολουθεί την τακτική No-Flag ως κινήσεις θεωρούνται οι αποκαλύψεις μπλοκ με κλικ. Σε ένα μεγαλύτερο ταμπλό είναι λογικό να χρειάζονται περισσότερες αποκαλύψεις, όπως παρατηρείται σε κάθε στήλη του πίνακα 4.6. Παρατηρώντας τις επιπτώσεις της πυκνότητας, όμως, γίνεται εμφανές πως αυξάνει σημαντικά τον αριθμό κινήσεων. Όπως αναφέρθηκε στην αρχή της ενότητας, αυτό συμβαίνει διότι οι κινήσεις έχουν μικρότερη πιθανότητα να αποκαλύψουν μεγαλύτερο μέρος του ταμπλό.

	0,12345	**0,15626**	**0.20625**
9x9	13.661	19.288	24.314
16x16	26.372	39.431	60.42
30x16	35.222	52.864	83.999

Πίνακας 4.6: Σύνολο κινήσεων για διαφορετικές πυκνότητες στις βασικές διαστάσεις ταμπλό.



Εικόνα 4.3: Επιπτώσεις της πυκνότητας και των διαστάσεων στον αριθμό των κινήσεων έως το τέλος της παρτίδας.

Όταν δεν υπάρχουν σίγουρες κινήσεις μετά την πιθανοτική ανάλυση του ταμπλό, απαιτούνται οι «τυχαίες» κινήσεις από τον επιλυτή. Όπως παρουσιάζεται στην Εικόνα 4.4a

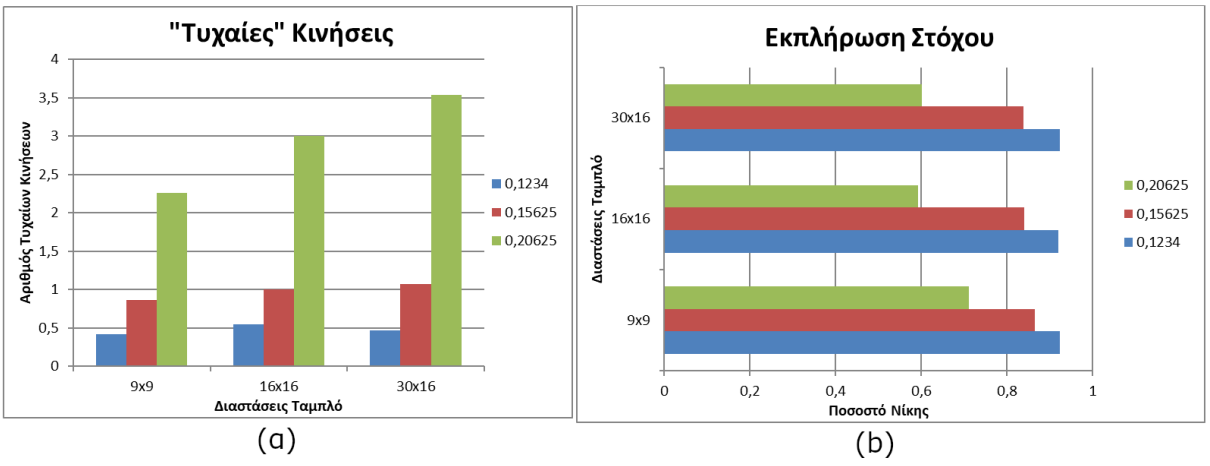
και στον πίνακα 4.7, η πυκνότητα των ναρκών οδηγεί σε περισσότερες καταστάσεις που απαιτούν τέτοιες κινήσεις. Στην αρχή της ενότητας έγινε αναφορά σε ζεύγη μπλοκ που είναι ισοπίθανα νάρκες (50-50) τα οποία ανήκουν σε αυτές τις καταστάσεις. Δεδομένου ότι, υπό έλλειψη σίγουρης κίνησης, ο επιλυτής διαλέγει προς αποκάλυψη το μπλοκ με τις χαμηλότερες πιθανότητες να είναι νάρκη, οι αποφάσεις με 50% πιθανότητα μένουν για το τέλος της παρτίδας. Φυσικά, ένας παίκτης θα αποφάσιζε την ίδια στιγμή ώστε, αν είναι νάρκη, να περάσει στην επόμενη παρτίδα. Αυτή δεν είναι η περίπτωση για τον επιλυτή της εργασίας καθώς ο στόχος του είναι η εύρεση του μεγαλύτερου δυνατού αριθμού ναρκών ακόμα και όταν χάνει μία παρτίδα. Η στρατηγική αυτή αποτυπώνεται στα στατιστικά του πίνακα 4.8 καθώς και στην εικόνα 4.4b.

	0,12345	**0,15626**	**0.20625**
9x9	0.415	0.866	2.26
16x16	0.546	1.005	3.002
30x16	0.47	1.072	3.532

Πίνακας 4.7: Σύνολο τυχαίων κινήσεων για διαφορετικές πυκνότητες στις βασικές διαστάσεις ταμπλό.

	0,12345	**0,15626**	**0.20625**
9x9	0.9235	0.8644	0.7114
16x16	0.9191	0.8392	0.5917
30x16	0.9226	0.8383	0.6007

Πίνακας 4.8: Ποσοστό εκπλήρωσης στόχου για διαφορετικές πυκνότητες στις βασικές διαστάσεις ταμπλό.



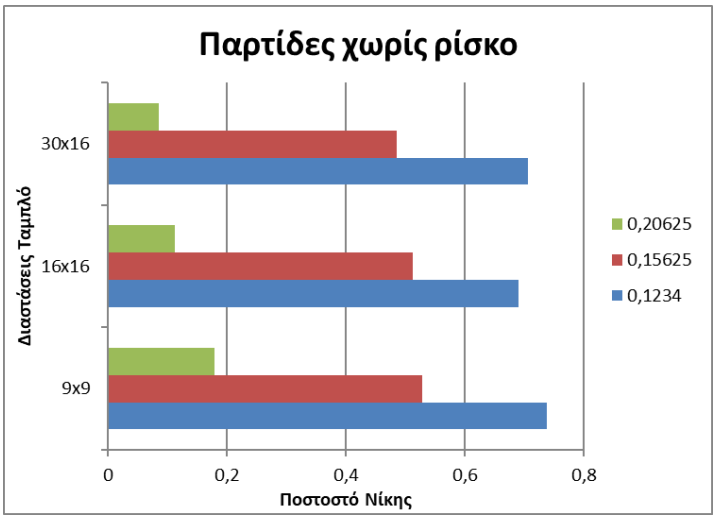
(a) (b)

Εικόνα 4.4: Επιπτώσεις της πυκνότητας και των διαστάσεων (a) στον αριθμό τυχαίων κινήσεων έως το τέλος της παρτίδας (b) στην ποσοστιαία εκπλήρωση του στόχου της εύρεσης των περισσότερων δυνατών ναρκών.

Έχοντας αναλύσει το μέσο αριθμό «τυχαίων» κινήσεων ανά παρτίδα, μπορεί να γίνει η εξαγωγή συμπερασμάτων σχετικά με τις παρτίδες χωρίς ρίσκο. Κατά τις παρτίδες αυτές έγινε χρήση βασικών στρατηγικών και επίλυσης ως CSP αλλά υπήρξαν μόνο στατιστικά σίγουρες κινήσεις. Μελετώντας τον πίνακα 4.8 και την οπτικοποίηση του στην Εικόνα 4.5, η διακύμανση του ποσοστού παρτίδων χωρίς ρίσκο μεταξύ της χαμηλότερης και υψηλότερης πυκνότητας είναι εμφανής.

	0,12345	**0,15626**	**0.20625**
9x9	0.738	0.528	0.18
16x16	0.69	0.513	0.112
30x16	0.706	0.486	0.085

Πίνακας 4.9: Ποσοστό παρτίδων χωρίς ρίσκο για διαφορετικές πυκνότητες στις βασικές διαστάσεις ταμπλό.



Εικόνα 4.5: Επιπτώσεις της πυκνότητας και των διαστάσεων στον αριθμό των παρτίδων χωρίς ρίσκο.

4.5 Σύνοψη Κεφαλαίου

Κατά το κεφάλαιο, διατυπώθηκαν οι μέθοδοι ελέγχου ορθής λειτουργίας σε διαφορετικές συσκευές και η διαδικασία συλλογής στατιστικών στοιχείων από το πειραματικό περιβάλλον. Εξετάστηκε η απόδοση και το ποσοστό επιτυχίας του επιλυτή ενώ έγινε και σύγκριση με άλλες εργασίες. Παράλληλα έγινε εμβάθυνση στην επιρροή της πυκνότητας και του μεγέθους του ταμπλό σε διάφορους παράγοντες της επίλυσης. Στο κεφάλαιο που ακολουθεί θα διατυπωθούν τα συμπεράσματα από την υλοποίηση της παρούσας εργασίας καθώς και πιθανές μελλοντικές επεκτάσεις.

Κεφάλαιο 5

Συμπεράσματα και μελλοντικές βελτιώσεις

Στο κεφάλαιο αυτό συνοψίζονται τα συμπεράσματα που εξήχθησαν κατά την εκπόνηση της διπλωματικής εργασίας. Επιπρόσθετα, παρουσιάζονται μελλοντικές βελτιώσεις που μπορούν να επεκτείνουν τις δυνατότητες του Αυτοματοποιημένου Επιλυτή και να του προσδώσουν μεγαλύτερη απόδοση.

5.1 Συμπεράσματα

Κατά την παρούσα διπλωματική εργασία, αναπτύχθηκε μια μοναδική προσέγγιση με στόχο την οπτική αναγνώριση και την επίλυση του Ναρκαλιευτή. Ο επιλυτής απέκλινε από τις μεθόδους που προϋπήρχαν, συνδυάζοντας τεχνικές μηχανικής όρασης για την εύρεση του ταμπλό στην οθόνη του χρήστη και την αναγνώριση της κατάστασης των επιμέρους μπλοκ. Έχοντας καλύψει απρόσκοπτα και αποδοτικά τις ανάγκες εισόδου, ο επιλυτής εφαρμόζει βασικούς ντετερμινιστικούς αλγορίθμους για να ξεδιαλύνει τις απλές καταστάσεις του ταμπλό ενώ μοντελοποιεί ως Προβλήματα Ικανοποίησης Περιορισμών τις πιο απαιτητικές. Αξιοποιώντας τον επιλυτή CP-SAT που βασίζεται στην υβριδική τεχνική LCG, εκμεταλλεύεται τα πλεονεκτήματα των μοντέλων υψηλού επιπέδου και της αυτόνομης αναζήτησης των SAT για την εύρεση πιθανών λύσεων. Με βάση τις πιθανές λύσεις που εξάγει το CP-SAT, προσδίδεται ακρίβεια στην επιλογή των επόμενων κινήσεων με την πιθανοτική ανάλυση του ταμπλό.

Όπως είναι εμφανές, ο Αυτοματοποιημένος Επιλυτής ενσωματώνει τα προτερήματα διαφόρων τεχνικών αλλά ταυτόχρονα επωμίζεται τους περιορισμούς που ενέχουν. Με τη χρήση μεθόδων ψηφιακής επεξεργασίας και ανάλυσης εικόνας, η είσοδος του επιλυτή εξασφαλίζεται με αισθητά βελτιωμένο χρόνο εκτέλεσης συγκριτικά με άλλες μεθόδους οπτικής αναγνώρισης. Δεδομένου ότι επιτρέπει την επίλυση κάθε παρτίδας δίχως πρόσβαση στον κώδικα του παιχνιδιού, πρόκειται για μια ευέλικτη λύση σε ένα πρόβλημα που παίρνει πολλές μορφές. Στον αντίποδα, η προσέγγιση αυτή απαιτεί μεγάλη ακρίβεια στη διαμόρφωση κάθε μοντέλου αναγνώρισης αντικειμένου, κάτι το οποίο συνήθως είναι αντιστρόφως ανάλογο με το εύρος των προδιαγραφών και των εκδόσεων του παιχνιδιού που μπορεί να υποστηρίξει. Κατά συνέπεια, τίθενται περιορισμοί σχετικά με τις ρυθμίσεις της οθόνης και

την έκδοση του Ναρκαλιευτή ώστε να συνάδουν με το μοντέλο που διατίθεται στην προγραμματιστική υλοποίηση.

Σε συνδυασμό με τη μοναδική μέθοδο για την παροχή εισόδου, ο Αυτοματοποιημένος Επιλυτής διατηρεί ένα υψηλό ποσοστό επιτυχίας σε αποδεκτό χρόνο επίλυσης. Για να επιτύχει αυτό, ενορχηστρώνει τις βασικές στρατηγικές επίλυσης και τις εναλλάσσει με τον επιλυτή CP-SAT μονάχα όταν είναι απόλυτα απαραίτητο. Αυτό συμβαίνει διότι η εύρεση πιθανών αναθέσεων μπορεί να περιέχει εκτενή χώρο αναζήτησης, κάτι που μεταφράζεται σε εκτεταμένο χρόνο CPU, όπως αποδείχθηκε κατά την πειραματική διαδικασία. Πιο αναλυτικά, ενώ ο χρόνος στις βασικές κλίμακες δυσκολίας είναι κατά πλειοψηφία αποδεκτός, οι μεγαλύτερες πυκνότητες ναρκών μαζί με μεγαλύτερα ταμπλό μπορεί να αποδειχτούν δυσεπίλυτη πρόκληση με αυξανόμενες χρονικές απαιτήσεις.

Κατά την πειραματική διαδικασία, έγινε εξαγωγή συμπερασμάτων με βάση τα στατιστικά στοιχεία από 6500 παρτίδες. Από αυτές τις παρτίδες παρατηρήθηκε η εκτεταμένη επιρροή της πυκνότητας των ναρκών σχεδόν σε όλες τις πτυχές της εκτέλεσης καθώς και ο συσχετισμός μεταξύ τους.

5.2 Μελλοντικές Επεκτάσεις

Με την υλοποίηση του Αυτοματοποιημένου Επιλυτή που παρουσιάστηκε κατά την εργασία, έγινε συνδυασμός σύγχρονων μεθόδων για την αντιμετώπιση της πρόκλησης του Ναρκαλιευτή. Χάρη σε αυτό, τέθηκαν βάσεις για μελλοντική επέκταση και βελτίωση των μεθόδων ώστε να υπάρχει ευρύτερη υποστήριξη τεχνικών προδιαγραφών αλλά και να καλύπτονται περισσότερες εκδόσεις του παιχνιδιού. Παράλληλα, οι βελτιώσεις έχουν ως στόχο την αύξηση της απόδοσης για την ανάληψη δυσκολότερων προκλήσεων (όπως μεγαλύτερα ταμπλό ή πυκνότητες) καθώς και την ελαχιστοποίηση των περιορισμών τόσο για το CP-SAT όσο και για την αναγνώριση του ταμπλό.

5.2.1 Εκτεταμένα Μοντέλα Αναγνώρισης

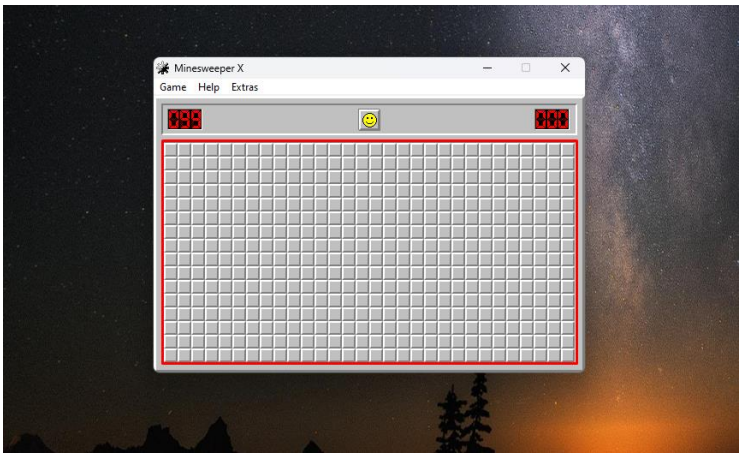
Δεδομένου της επιτυχίας του Ναρκαλιευτή, βρίσκονται σε κυκλοφορία πολλαπλές εκδόσεις που βασίζονται στο πρωτότυπο. Καθώς οι κανόνες και η αλληλεπίδραση με το παιχνίδι ακολουθούν τα ίδια πρότυπα, η ειδοποιός διαφορά βρίσκεται κυρίως στην εμφάνιση του παιχνιδιού και στην απεικόνιση των διαφόρων πτυχών του με πιο σύγχρονα μέσα.

Έχοντας βασιστεί στην εμφάνιση και στις προδιαγραφές της έκδοσης των Windows XP, το μοντέλο αναγνώρισης της εργασίας είναι ειδικό και απαιτεί από το χρήστη να έχει εξασφαλίσει συγκεκριμένες ρυθμίσεις πριν την εκτέλεση.

Για την αναγνώριση άλλων εκδόσεων, θα μπορούσε, για κάθε μία, να αναλυθεί εκ νέου το ταμπλό, οι αριθμοί και οι συνθήκες τερματισμού παρτίδας για τη διαμόρφωση πολλαπλών μοντέλων. Με τη χρήση αυτών των μοντέλων, ο επιλυτής θα αναγνωρίζει στην αρχή της παρτίδας ποια έκδοση βρίσκεται στην οθόνη και θα εφαρμόζει το κατάλληλο μοντέλο κατά την επίλυση. Μία πιο σύνθετη προσέγγιση, που δε θα απαιτεί επιπλέον χρόνο στην αρχή της επίλυσης, είναι η μελέτη και η δόμηση ενός γενικευμένου μοντέλου αναγνώρισης που θα εντοπίζει κάθε ταμπλό ανεξάρτητα από την έκδοση. Αξίζει να σημειωθεί πως και σε αυτή την προσέγγιση θα πρέπει να γίνει εκ νέου μελέτη των εντάσεων των αριθμών, ώστε να περιλαμβάνουν ευρύτερες τιμές. Για την αφαίρεση του περιορισμού διαστάσεων 16x16 εικονοστοιχείων για κάθε μπλοκ που έχει το μοντέλο της εργασίας μπορούν να εφαρμοστούν περαιτέρω υπολογισμοί στην αναγνώριση του ταμπλό. Αξιοποιώντας τις γραμμές που εξάγει ο μετασχηματισμός Hough, μπορούν να εντοπιστούν τα σημεία στα οποία τέμνονται ως οι γωνίες των μπλοκ. Με βάση αυτά τα σημεία και την απόσταση που τα χωρίζει, μπορούν να εντοπιστούν δυναμικά οι διαστάσεις των μπλοκ. Η επέκταση αυτή θα ακύρωνε ενδεχομένως και την ανάγκη για τήρηση συγκεκριμένης ανάλυσης οθόνης.

5.2.2 Ανάληψη Παρτίδας σε εξέλιξη

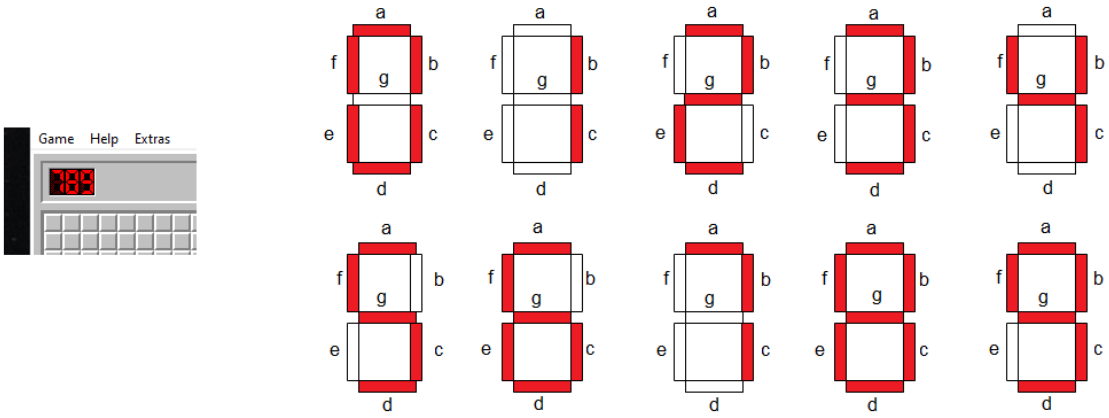
Η μέθοδος εύρεσης και απομόνωσης του ταμπλό που ακολουθεί ο επιλυτής της εργασίας απαιτεί να είναι κλειστά όλα τα μπλοκ ώστε να μετασχηματίζονται επαρκείς γραμμές. Έχοντας σαν δεδομένο την τοποθεσία και τις διαστάσεις του ταμπλό, ο επιλυτής είναι σε θέση να αναλάβει οποιαδήποτε παρτίδα. Για την ανάληψη κάποιας παρτίδας σε εξέλιξη, υπάρχουν δύο πιθανές επεκτάσεις. Κατά την πρώτη, θα μπορούσε να υλοποιηθεί μηχανισμός μεταφοράς και απόθεσης (drag and drop) για την επιλογή του πεδίου ενδιαφέροντος που είναι το ταμπλό όπως αποτυπώνεται στην Εικόνα 5.1. Εναλλακτικά, μπορούν να βρεθούν διαφορετικές παράμετροι για το μετασχηματισμό Hough σε συνδυασμό με εξίσου αποτελεσματικό φιλτράρισμα για να παραμένουν μόνο τα περιγράμματα του ταμπλό. Αδιαμφισβήτητα, η δεύτερη προσέγγιση κρύβει πολλαπλές προκλήσεις αν συνδυαστεί με ένα γενικευμένο μοντέλο.



Εικόνα 5.1: Μηχανισμός Drag and Drop με κόκκινη επισήμανση του πλαισίου για την επιλογή του ταμπλό.

5.2.3 Αναγνώριση επταμερούς ένδειξης αριθμών

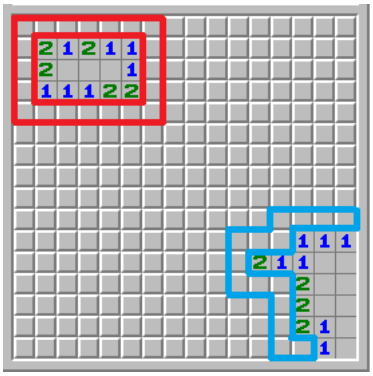
Στην άνω αριστερή πλευρά του ταμπλό, υπάρχει μια ένδειξη για τον αριθμό των ναρκών που βρίσκονται στο πεδίο των μπλοκ. Η ένδειξη αυτή έχει μορφή επταμερούς ένδειξης αριθμών (7-digit-display) με κόκκινο χρώμα. Με μεθόδους κατωφλίωσης και εύρεσης περιγραμμάτων μπορούν να βρεθούν και να απομονωθούν τα επτά μέρη της ένδειξης. Με βάση την ενεργοποίηση (ή μη) κάποιου μέρους της ένδειξης αλλάζει και η ερμηνεία του αριθμού που στοχεύει να αποτυπώσει. Για παράδειγμα, στην Εικόνα 5.1 η ενεργοποίηση όλων των μερών αποτυπώνει τον αριθμό 8 ενώ αν απενεργοποιηθεί το μέρος e δημιουργείται ο αριθμός 9. Δημιουργώντας μία λίστα με τον τρόπο που εκφράζεται κάθε αριθμός, δίνεται η δυνατότητα να εξαλειφθεί η ανάγκη για εισαγωγή ναρκών στο αριθμητικό πεδίο από το χρήστη σε περιπτώσεις προσαρμοσμένης δυσκολίας.



Εικόνα 5.2: Επταμερής ένδειξη αριθμών και η έκφραση κάθε αριθμού ως συνδυασμός ενεργοποιήσεων μερών.

5.2.4 Βελτίωση απόδοσης

Όπως αποδείχθηκε κατά την πειραματική διαδικασία, η μεγαλύτερη πυκνότητα αυξάνει τις επιλύσεις ως CSP και κατά συνέπεια το χρόνο CPU. Αυτό σημαίνει πως μπορεί να γίνει βελτίωση των μοντέλων για την ανάληψη δυσκολότερων καταστάσεων με περισσότερους περιορισμούς και εξεταζόμενα μπλοκ. Η βελτίωση αυτή μπορεί να επέλθει με την ομαδοποίηση των περιορισμών με βάση τη συνάφεια για την παράλληλη επίλυση με πολλαπλά μοντέλα και επιμέρους ενεργοποιήσεις του CP-SAT σε νήματα. Στην εικόνα 5.3 παρουσιάζεται η ομαδοποίηση σαν «αλυσίδες» περιορισμών που μοιράζονται μπλοκ άκρης. Αξίζει να σημειωθεί πως, κατά το στάδιο της προεπεξεργασίας του CP-SAT, γίνεται εμμέσως παρόμοια διεργασία με βάση την αλληλεπίδραση των μηχανισμών διάδοσης αλλά ενδέχεται να υπάρχει περιθώριο βελτίωσης με την προτεινόμενη επέκταση.



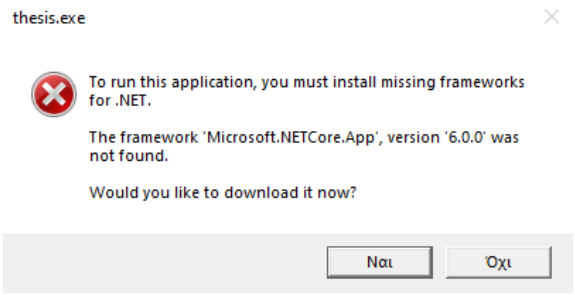
Εικόνα 5.3: Ομαδοποίηση ως αλυσίδες περιορισμών που μοιράζονται μπλοκ άκρης και επιλύονται ως ξεχωριστά μοντέλα CSP από ενεργοποιήσεις του CP-SAT σε επιμέρους νήματα.

Τέλος, δεδομένου του μεγέθους των αριθμών που διαχειρίζεται στις χείριστες περιπτώσεις ο επιλυτής, για την επέκταση του σε μεγαλύτερες διαστάσεις και πυκνότητες ναρκών πρέπει να εφαρμοστεί ο αλγόριθμος Karatsuba [58]. Ο αλγόριθμος αυτός προσφέρει οφέλη όσο αφορά τον υπολογισμό παραγοντικών αριθμών και θα επιταχύνει τον υπολογισμό των συνδυασμών για κάθε μπλοκ σε μία απαιτητική κατάσταση.

Παραρτήματα

Εγκατάσταση Αυτοματοποιημένου Επιλυτή

Ο Αυτοματοποιημένος Επιλυτής έχει υλοποιηθεί ως εφαρμογή Windows Forms με στόχο τη χρήση στο λειτουργικό σύστημα Microsoft Windows. Η εφαρμογή μπορεί να εκτελεστεί μέσω δυαδικού εκτελέσιμου αρχείου (.exe) εφόσον βρίσκεται στον ίδιο φάκελο με όλα τα δυαδικά αρχεία/βιβλιοθήκες τύπου DLL (Dynamic-Link Library). Καθώς βασίζεται στη δομή .NET 6.0, είναι απαραίτητο να υπάρχει εγκατεστημένη στο σύστημα του χρήστη. Σε περίπτωση που δεν εντοπιστεί κατά την εκκίνηση (και για τη διευκόλυνση του χρήση), εμφανίζεται προτροπή ώστε, αν το επιθυμεί, να ακολουθήσει το σύνδεσμο προς την επίσημη ιστοσελίδα της Microsoft [Εικόνα Π.1]. Στη σελίδα αυτή, αρκεί να επιλέξει προς κατέβασμα την έκδοση .NET Runtime που απευθύνεται στην εκτέλεση εφαρμογών και να ακολουθήσει τον οδηγό εγκατάστασης. Αφού ολοκληρωθεί, ο επιλυτής είναι έτοιμος να εκτελεστεί.



Εικόνα Π.1: Προτροπή για κατέβασμα του .NET 6.0.

Για την ορθή εκτέλεση του επιλυτή, ο χρήστης οφείλει να έχει διαθέσιμη την έκδοση Windows XP του Ναρκαλιευτή και να ακολουθήσει τις προδιαγραφές οι οποίες συνάδουν με το μοντέλο αναγνώρισης του. Οι ρυθμίσεις αυτές είναι οι εξής: (1) 1920x1080 ανάλυση οθόνης, (2) 100% κλίμακα ανάλυσης, (3) απενεργοποίηση χρωματικών φίλτρων ή άλλων εφαρμογών που τροποποιούν την αποτύπωση του παιχνιδιού. Οποιαδήποτε παρέκκλιση από τις προτεινόμενες ρυθμίσεις θα έχει αρνητικές επιπτώσεις στην ακρίβεια της αναγνώρισης του ταμπλό.

Βιβλιογραφία

[1] Moody Rebecca, “Screen Time Statistics: Average Screen Time in US vs. the Rest of the World.,” *Comparitech*, 2022. https://www.comparitech.com/tv-streaming/screen-time-statistics/ (accessed Jun. 17, 2022).

[2] J. Juul, “The repeatedly lost art of studying games,” *Game Studies*, vol. 1, no. 1, 2001.

[3] M. Newborn, *Kasparov versus Deep Blue: Computer chess comes of age*. Springer Science & Business Media, 2012.

[4] D. A. Ferrucci, “Introduction to ‘this is watson,’” *IBM Journal of Research and Development*, vol. 56, no. 3.4, p. 1, 2012.

[5] V. Mnih *et al.*, “Playing atari with deep reinforcement learning,” *arXiv preprint arXiv:1312.5602*, 2013.

[6] “AlphaGO,” *DeepMind*. https://www.deepmind.com/research/highlighted-research/. (accessed Jun. 17, 2022).

[7] “OpenAI Five,” *OpenAI Five*. https://openai.com/five/ (accessed Jun. 17, 2022).

[8] O. Vinyals *et al.*, “Grandmaster level in StarCraft II using multi-agent reinforcement learning,” *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[9] B. Shestakofsky, “Working algorithms: Software automation and the future of work,” *Work and Occupations*, vol. 44, no. 4, pp. 376–423, 2017.

[10] Kaye R, “Minesweeper is NP-Complete,” 2000.

[11] I. Stewart, “Million-Dollar Minesweeper,” 2000. [Online]. Available: http://www.minesweeper.info/articles/

[12] R. Kaye, “Infinite versions of minesweeper are Turing complete,” 2000. [Online]. Available: http://www.mat.bham.ac.uk/R.W.Kaye

[13] K. Pedersen, “The complexity of Minesweeper and strategies for game playing,” 2003.

[14] M. de Bondt, “The computational complexity of Minesweeper,” Apr. 2012, [Online]. Available: http://arxiv.org/abs/1204.4659

[15] A. Adamatzky, “How Cellular Automaton Plays Minesweeper,” 1985.

[16] D. Kamenetsky and C. H. Teo, “Graphical Models for Minesweeper Project Report,” 2007.

[17] S. Golan, “Minesweeper on graphs,” *Appl Math Comput*, vol. 217, no. 14, pp. 6616–6623, 2011.

[18] P. Nakov and Z. Wei, “MINESWEEPER, #MINESWEEPER,” 2003.

[19] L. P. Castillo and S. Wrobel, “Learning minesweeper with multirelational learning,” in *INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE*, 2003, vol. 18, pp. 533–540.

[20] M. Vomlelová and J. Vomlel, “Applying Bayesian networks in the game of Minesweeper,” 2009. [Online]. Available: http://kti.mff.cuni.cz/~marta/http://www.utia.cas.cz/vomlel

[21] O. Buffet, C. S. Lee, W. T. Lin, and O. Teytuad, “Optimistic heuristics for mineSweeper,” *Smart Innovation, Systems and Technologies*, vol. 20, pp. 199–207, 2013, doi: 10.1007/978-3-642-35452-6_22.

[22] K. Bayer, J. Snyder, and B. Y. Choueiry, “An Interactive Constraint-Based Approach to Minesweeper,” 2006. [Online]. Available: www.aaai.org

[23] R. Collet, “Playing the Minesweeper with Constraints,” 2004.

[24] C. Studholme, “Minesweeper as a Constraint Satisfaction Problem,” 2000.

[25] M. Legendre, K. Hollard, O. Buffet, and A. Dutech, “MineSweeper: Where to Probe?,” 2012. [Online]. Available: https://hal.inria.fr/hal-00723550

[26] J. Tu, T. Li, S. Chen, C. Zu, and Z. Gu, “Exploring Efficient Strategies for Minesweeper,” 2017. [Online]. Available: www.aaai.org

[27] S. Golan, “Minesweeper strategy for one mine,” *Applied Mathematics and Computation*, vol. 232, pp. 292–302, 2014.

[28] Beccera David J., “Algorithmic Approaches to Playing Minesweeper,” 2015. [Online]. Available: http://nrs.harvard.edu/urn-3:HUL.InstRepos:14398552

[29] B. Packard, “Solving a Minesweeper Board By Visually Parsing It,” 2021.

[30] “Fastest combined completion time of Minesweeper,” *Guinness World Records*. https://www.guinnessworldrecords.com/world-records/fastest-completions-of-minesweeper (accessed Jun. 17, 2022).

[31] R. Szeliski, *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[32] H. Sidenbladh, M. J. Black, and D. J. Fleet, “Stochastic tracking of 3D human figures using 2D image motion,” in *European conference on computer vision*, 2000, pp. 702–718.

[33] J. Sivic, C. L. Zitnick, and R. Szeliski, “Finding People in Repeated Shots of the Same Scene.,” in *BMVC*, 2006, vol. 2, p. 3.

[34] T. E. de Campos, B. R. Babu, and M. Varma, “Character recognition in natural images.,” *VISAPP (2)*, vol. 7, no. 2, 2009.

[35] D. Mery, “Computer vision for X-Ray testing,” *Switzerland: Springer International Publishing*, vol. 10, pp. 973–978, 2015.

[36] D. Sankowski and J. Nowakowski, *Computer vision in robotics and industrial applications*, vol. 3. World Scientific, 2014.

[37] M. Daily, S. Medasani, R. Behringer, and M. Trivedi, “Self-driving cars,” *Computer (Long Beach Calif)*, vol. 50, no. 12, pp. 18–23, 2017.

[38] R. C. Gonzalez, *Digital image processing*. Pearson education india, 2009.

[39] T. Kumar and K. Verma, “A Theory Based on Conversion of RGB image to Gray image,” *International Journal of Computer Applications*, vol. 7, no. 2, pp. 7–10, 2010.

[40] E. S. Gedraite and M. Hadad, “Investigation on the effect of a Gaussian Blur in image filtering and segmentation,” in *Proceedings ELMAR-2011*, 2011, pp. 393–396.

[41] J. Canny, “A computational approach to edge detection,” *IEEE Transactions on pattern analysis and machine intelligence*, no. 6, pp. 679–698, 1986.

[42] Z. Xu, X. Baojie, and W. Guoxin, “Canny edge detection based on Open CV,” in *2017 13th IEEE international conference on electronic measurement & instruments (ICEMI)*, 2017, pp. 53–56.

[43] E. R. Davies, *Computer and machine vision: theory, algorithms, practicalities*. Academic Press, 2012.

[44] D. Forsyth and J. Ponce, *Computer vision: A modern approach*. Prentice hall, 2011.

[45] Y. Zhu, R. Urtasun, R. Salakhutdinov, and S. Fidler, “segdeepm: Exploiting segmentation and context in deep neural networks for object detection,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 4703–4711.

[46] R. O. Duda and P. E. Hart, “Use of the Hough transformation to detect lines and curves in pictures,” *Commun ACM*, vol. 15, no. 1, pp. 11–15, 1972.

[47] P. Arbelaez, M. Maire, C. Fowlkes, J. Malik, and P. Arbeí Aez, “Contour Detection and Hierarchical Image Segmentation,” 2010. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-17.html

[48] R. M. Karp, “Reducibility among combinatorial problems,” in *Complexity of computer computations*, Springer, 1972, pp. 85–103.

[49] D. S. Johnson, “A catalog of complexity classes,” in *Algorithms and complexity*, Elsevier, 1990, pp. 67–161.

[50] J. Hartmanis and R. E. Stearns, “On the computational complexity of algorithms,” *Transactions of the American Mathematical Society*, vol. 117, pp. 285–306, 1965.

[51] T. Baker, J. Gill, and R. Solovay, “Relativizations of the P=?NP question,” *SIAM Journal on computing*, vol. 4, no. 4, pp. 431–442, 1975.

[52] L. Fortnow, “The status of the P versus NP problem,” *Commun ACM*, vol. 52, no. 9, pp. 78–86, 2009.

[53] G. Meurant, *Algorithms and Complexity*. Elsevier Science, 2014. [Online]. Available: https://books.google.gr/books?id=6WriBQAAQBAJ

[54] V. Arvind and P. P. Kurur, “Graph Isomorphism is in SPP,” *Information and Computation*, vol. 204, no. 5, pp. 835–852, 2006, doi: 10.1016/j.ic.2006.02.002.

[55] C. H. Papadimitriou and U. v Vazirani, “On two geometric problems related to the travelling salesman problem,” *Journal of Algorithms*, vol. 5, no. 2, pp. 231–246, 1984.

[56] P. L. Jensen, “Integer factorization,” *University of Copenhagen*, 2005.

[57] R. Beigel, “Bounded queries to SAT and the Boolean hierarchy,” *Theor Comput Sci*, vol. 84, no. 2, pp. 199–223, 1991.

[58] A. Weimerskirch and C. Paar, “Generalizations of the Karatsuba algorithm for efficient implementations,” *Cryptology ePrint Archive*, 2006.

[59] J. Renegar, “Linear programming, complexity theory and elementary functional analysis,” *Mathematical Programming*, vol. 70, no. 1, pp. 279–351, 1995.

[60] M. Agrawal, N. Kayal, and N. Saxena, “PRIMES is in P,” 2004.

[61] M. R. Garey and D. S. Johnson, “Computers and intractability,” *A Guide to the*, 1979.

[62] S. A. Cook, “The complexity of theorem-proving procedures,” in *Proceedings of the third annual ACM symposium on Theory of computing*, 1971, pp. 151–158.

[63] H. Kellerer, U. Pferschy, and D. Pisinger, “Introduction to NP-Completeness of knapsack problems,” in *Knapsack problems*, Springer, 2004, pp. 483–493.

[64] M. R. Garey and D. S. Johnson, “The complexity of near-optimal graph coloring,” *Journal of the ACM (JACM)*, vol. 23, no. 1, pp. 43–49, 1976.

[65] A. Itai, C. H. Papadimitriou, and J. L. Szwarcfiter, “Hamilton paths in grid graphs,” *SIAM Journal on Computing*, vol. 11, no. 4, pp. 676–686, 1982.

[66] C. Lecoutre, *Constraint Networks: Targeting Simplicity for Techniques and Algorithms*. John Wiley & Sons, 2013.

[67] A. A. Bulatov, “A dichotomy theorem for nonuniform CSPs,” in *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, 2017, pp. 319–330.

[68] K. Claessen, N. Een, M. Sheeran, and N. Sorensson, “SAT-solving in practice,” in *2008 9th International Workshop on Discrete Event Systems*, 2008, pp. 61–67.

[69] J. H. Liang, V. Ganesh, E. Zulkoski, A. Zaman, and K. Czarnecki, “Understanding VSIDS branching heuristics in conflict-driven clause-learning SAT solvers,” in *Haifa Verification Conference*, 2015, pp. 225–241.

[70] O. Ohrimenko, P. J. Stuckey, and M. Codish, “Propagation via lazy clause generation,” *Constraints*, vol. 14, no. 3, pp. 357–391, 2009.

[71] T. Feydy and P. J. Stuckey, “Lazy clause generation reengineered,” in *International Conference on Principles and Practice of Constraint Programming*, 2009, pp. 352–366.

[72] Stuckey P, “Search is Dead Long Live Proof,” 2013.

[73] “C# Documentation,” *Microsoft*. https://docs.microsoft.com/en-us/dotnet/csharp/ (accessed Jun. 17, 2022).

[74]. “NET Documentation,” *Microsoft*. https://docs.microsoft.com/en-us/dotnet/fundamentals/ (accessed Jun. 17, 2022).

[75] A. Kaehler and G. Bradski, *Learning OpenCV 3: computer vision in C++ with the OpenCV library*. “ O’Reilly Media, Inc.,” 2016.

[76] Laurent Perron, “OR-Tools | Google Developers.,” *Google*. https://developers.google.com/optimization/ (accessed Jun. 17, 2022).