ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Σχεδίαση και Υλοποίηση ενός Αυτόνομου Πράκτορα για μία Παραλλαγή του Παιχνιδιού της Ναυμαχίας

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ψύχας Χριστόφορος

Εξεταστική Επιτροπή

Αναπληρωτής Καθηγητής Μιχαήλ Λαγουδάκης (επιβλέπων)

Αναπληρωτής Καθηγητής Γεώργιος Χαλκιαδάκης

Καθηγητής Αντώνιος Δεληγιαννάκης

Χανιά, Μάρτιος 2022

TECHNICAL UNIVERSITY OF CRETE

SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

Design and Implementation of Autonomous Agent for a Variant of the Battleship Game

DIPLOMA THESIS

Psychas Christoforos

Thesis Committee

Associate Professor Michail Lagoudakis (supervisor)

Associate Professor Georgios Chalkiadakis

Professor Antonios Deligiannakis

Chania, March 2022

Περίληψη

Η Τεχνητή Νοημοσύνη αναφέρεται στην σχεδίαση και υλοποίηση υπολογιστικών συστημάτων, τα οποία μιμούνται την ανθρώπινη συμπεριφορά και επιδεικνύουν στοιχεία ευφυΐας. Αυτές οι οντότητες είναι γνωστές και ως «ευφυείς πράκτορες». Η ιδέα σχεδιασμού ενός πράκτορα, ο οποίος θα μπορούσε να νικήσει τον άνθρωπο σε ένα παιχνίδι, ήταν πάντα συναρπαστική. Η παρούσα διπλωματική εργασία εστιάζει στη σχεδίαση και ανάπτυξη ενός αυτόνομου πράκτορα για μια παραμετρική παραλλαγή του επιτραπέζιου παιχνιδιού «Ναυμαχία», όπου το μέγεθος του πίνακα, καθώς επίσης το πλήθος και το μέγεθος των πλοίων μπορούν να ορισθούν από τον χρήστη. Στο πλαίσιο αυτής της εργασίας, αρχικά σχεδιάσαμε το γραφικό περιβάλλον του παιχνιδιού, χρησιμοποιώντας τη γλώσσα προγραμματισμού Java και τις βιβλιοθήκες της, ώστε να οπτικοποιούνται οι προτεινόμενες στρατηγικές του πράκτορά μας, αλλά και να δίνεται η δυνατότητα σε πραγματικούς χρήστες να παίξουν το παιχνίδι, λαμβάνοντας πρόσθετη πληροφόρηση μέσω των στρατηγικών. Η προσέγγισή μας βασίστηκε στον ορισμό προοδευτικά βελτιωμένων στρατηγικών, υιοθετώντας κάθε φορά νέους ευριστικούς κανόνες βασισμένους στην ανθρώπινη διαίσθηση και βασικά στοιχεία θεωρίας πιθανοτήτων. Τα αποτελέσματα ήταν αρκετά ικανοποιητικά, καθώς ο πράκτορας μας με χρήση των καλύτερων στρατηγικών απέδωσε πολύ καλά συγκριτικά ακόμη και με πραγματικούς χρήστες, πετυχαίνοντας τη βύθιση και των πέντε πλοίων του παιχνιδιού σε μόλις 44 βολές κατά μέσο όρο σε πίνακα 10x10. Τα αποτελέσματα επιβεβαιώνονται με ενδελεχή πειραματισμό, τόσο στην κλασσική εκδοχή σε πίνακα 10x10, όσο και στην πιο δύσκολη εκδοχή σε πίνακα 20x20, ακόμη και απέναντι σε έξυπνες τοποθετήσεις πλοίων.

Abstract

Artificial Intelligence refers to the design and implementation of computer systems that mimic human behavior and exhibit elements of intelligence. These entities are also known as "intelligent agents". The idea of designing an agent who could beat a human in a game has always been fascinating. This diploma thesis focuses on the design and development of an autonomous agent for a parametric variant of the board game "Battleship", where the size of the board, as well as the number and size of ships can be defined by the user. As part of this work, we initially designed the graphical interface of the game, using the Java programming language and its libraries, in order to visualize our agent's proposed strategies, but also to enable real users to play the game, receiving additional information through our strategies. Our approach was based on the definition of progressively improved strategies, each time adopting new heuristic rules based on human intuition and basic elements of probability theory. The results were quite satisfactory, as our agent using the best strategies performed very well compared even to real users, succeeding in sinking all five ships of the game in just 44 shots on average on a 10x10 board. The results are confirmed by thorough experimentation, both in the classic version of a 10x10 board, and in the more difficult version of a 20x20 board, even in the face of clever ship placements.

Αφιερωμένο σε όλους αυτούς που πίστεψαν σε μένα, και συνέχισαν να πιστεύουν ως το τέλος.

«Η επιτυχία δεν είναι τελική. Η αποτυχία δεν είναι θανάσιμη. Είναι το θάρρος να συνεχίζεις αυτό που μετράει.»

Winston Churchill

Ευχαριστίες

Η παρούσα διπλωματική εργασία πραγματοποιήθηκε στο Εργαστήριο Προγραμματισμού και Τεχνολογίας Ευφυών Υπολογιστικών Συστημάτων της Σχόλης Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Πολυτεχνείου Κρήτης, υπό την επίβλεψη του Αν. Καθηγητή κ. Μιχαήλ Λαγουδάκη.

Πρωτίστως, θα ήθελα να ευχαριστήσω θερμά τον καθηγητή μου, κ. Μιχαήλ Λαγουδάκη, για την αποδοχή και την εμπιστοσύνη που μου έδειξε όταν τον είχα πλησιάσει για να λάβω το θέμα της διπλωματικής εργασίας. Επίσης, τον ευχαριστώ για την απίστευτη υπομονή που έδειξε όλα αυτά τα χρόνια, καθώς βρισκόμουν για προσωπικούς λόγους μόνιμα εκτός της πόλης των Χανίων, αλλά και την καθοδήγηση του, χάρη στην οποία τελικώς ολοκληρώθηκε η παρούσα διπλωματική εργασία.

Στη συνέχεια, θα ήθελα να ευχαριστήσω την Δανάη Θανοπούλου, η οποία με υποστήριξε ψυχολογικώς τα τελευταία 2.5 χρόνια, χωρίς την επαγγελματική βοήθεια της οποίας πιθανώς να μην είχα πάρει απόφαση να αντιμετωπίσω πολλές φοβίες, μερικές από τις οποίες αφορούσαν και την εκπόνηση της εργασίας αυτής.

Έπειτα, οφείλω να ευχαριστήσω τους φίλους μου που στάθηκαν αρωγοί στην προσπάθεια μου όλα αυτά τα χρόνια, και ιδιαίτερα στον Ιωάννη Προεστάκη για την πολύτιμη βοήθεια του όταν σπουδάζαμε μαζί στα Χανιά, με τον οποίο είχαμε μία άψογη συνεργασία στα φοιτητικά μας χρόνια και με τον οποίο κέρδισα μία πολύτιμη φιλία. Επίσης, δε θα μπορούσα να παραλείψω να ευχαριστήσω φυσικά ίσως τον πιο κοντινό μου άνθρωπο, την Έλενα Σωτήρη, η οποία στάθηκε δίπλα μου ως βράχος καθ' όλη τη διάρκεια της εργασίας, και χάρη στην οποία δεν έχασα ποτέ το θάρρος και την πίστη μου ότι θα ολοκληρώσω την εργασία.

Τέλος, θα ήθελα να ευχαριστήσω την οικογένεια μου για την υποστήριξη και την συμπαράσταση που έδειξαν καθ' όλη τη διάρκεια των σπουδών μου. Ιδιαίτερα, θα ήθελα συγκεκριμένα να ευχαριστήσω τη μητέρα μου, η οποία ήταν πάντα εκεί για εμένα, και τον αδερφό μου, Δημήτριο Ψύχα, που οι πρόσφατες επιτυχίες στην διδακτορική του διατριβή και η απόκτηση του τίτλου του Δόκτορα με ενέπνευσαν στο τελικό στάδιο συγγραφής της διπλωματικής μου εργασίας, και οι βαθυστόχαστες συμβουλές του στην εργασία οδηγήσαν σε ένα πιο άρτιο αποτέλεσμα.

Σας ευχαριστώ όλους

Ψύχας Χριστόφορος

Πίνακας Περιεχομένων

1 Εισαγωγή

1.1 Εισαγωγή διπλωματικής εργασίας

Η ιδέα σχεδιασμού μιας μηχανής, η οποία θα είναι σε θέση να νικήσει τον άνθρωπο σε ένα παιχνίδι, ήταν πάντα συναρπαστική. Ήδη από τον 18ο αιώνα, ο Baron Wolfgang von Kempelen παρουσίασε το Maezal Chess Automaton (γνωστό στην Ευρώπη ως The Turk), το οποίο ήταν ουσιαστικά ένα είδος "αυτόματου" σκακιού, το οποίο κατάφερε να νικήσει παίκτες από όλο τον κόσμο, του ίδιου του Ναπολέοντα συμπεριλαμβανομένου, όπως και τον Benjamin Franklin. Είναι αυτονόητο ότι μια τέτοια προσπάθεια συγκέντρωσε το ενδιαφέρον του κόσμου, όπου και αν εμφανίστηκε δημόσια. Βέβαια, αργότερα αποκαλύφθηκε ότι όλο το εγχείρημα ήταν μια περίτεχνη απάτη, αφού το «αυτόματο» ήταν στην πραγματικότητα μια μηχανική ψευδαίσθηση, η οποία κατευθυνόταν από έναν κρυφό εμπειρογνώμονα στο συγκεκριμένο παιχνίδι.

Αυτό προοδευτικά εξελίχθηκε σε αυτό που ονομάζουμε Τεχνητή Νοημοσύνη, στο οποίο συντέλεσε η εξέλιξη της επιστήμης των υπολογιστών, η οποία έκανε μεγάλα βήματα μέσα από τη μελέτη των υπολογιστών. Με τον όρο Τεχνητή Νοημοσύνη αναφερόμαστε στον κλάδο της επιστήμης των υπολογιστών, ο οποίος ασχολείται με τη σχεδίαση και την υλοποίηση διάφορων υπολογιστικών συστημάτων, τα οποία έχουν την ικανότητα να μιμούνται στοιχεία της ανθρώπινης συμπεριφοράς, και έχουν και κάποια στοιχεία ευφυΐας, όπως μάθηση, προσαρμοστικότητα, εξαγωγή συμπερασμάτων, κατανόηση από συμφραζόμενα, επίλυση προβλημάτων, με την έννοια ότι ο υπολογιστής προσαρμόζει τις κινήσεις του, ανάλογα με την εξέλιξη του παιχνιδιού. Ο John McCarthy όρισε τον τομέα αυτόν ως «επιστήμη και μεθοδολογία της δημιουργίας νοούντων μηχανών». Η Τεχνητή Νοημοσύνη αποτελεί ένα πεδίο, στο οποίο συναντώνται η επιστήμη των υπολογιστών, της ψυχολογίας, της φιλοσοφίας, της νευρολογίας, της γλωσσολογίας, και της επιστήμης των μηχανικών, με σκοπό όχι μόνο τη κατανόηση, αλλά και τη σύνθεση μιας ευφυούς συμπεριφοράς, η οποία θα χαρακτηρίζεται από στοιχεία, όπως είναι η συλλογιστική, η μάθηση, και η προσαρμογή στο περιβάλλον και γενικότερα σε κάθε κατάσταση. Αυτές οι οντότητες ή κατασκευές, οι οποίες χαρακτηρίζονται από κάποιο επίπεδο νοημοσύνης, καλούνται «ευφυείς πράκτορες».

Οι πράκτορες είναι ουσιαστικά διάφορα υπολογιστικά συστήματα, τα οποία δρουν σε ένα πολύπλοκο περιβάλλον και είναι σε θέση να το αντιλαμβάνονται και να δρουν αυτόνομα εντός αυτού, επιτυγχάνοντας ένα σύνολο από στόχους για τους οποίους έχουν κατασκευαστεί. Σε αυτήν την προσέγγιση της Τεχνητής Νοημοσύνης αυτό που μας ενδιαφέρει είναι ο πράκτορας να είναι

ορθολογικός (rational), με την έννοια ότι η ιδανική κατάσταση είναι να ενεργεί με τρόπο τέτοιο, ώστε να επιτυγχάνει σε κάθε περίπτωση το καλύτερο δυνατό αποτέλεσμα ή, όταν αυτό δεν είναι εφικτό λόγω αβεβαιότητας, το καλύτερο αναμενόμενο αποτέλεσμα. Τέτοια περιβάλλοντα είναι τα παιχνίδια, τα οποία, από την σκοπιά της επιστήμης, είναι ενδιαφέροντα. Αν και οι κανόνες και οι ορισμοί τους μπορεί να είναι απλοί, τα παιχνίδια συχνά είναι πολύπλοκα. Τα διάφορα παιχνίδια αποτελούν ένα πεδίο έρευνας της Τεχνητής Νοημοσύνης από την πρώτη στιγμή που ο άνθρωπος ασχολήθηκε με αυτή. Ο λόγος που γίνεται μεγάλη χρήση τους είναι ότι τα περισσότερα από αυτά είναι καλά ορισμένα και έχουν ξεκάθαρους κανόνες, ενώ ένα ακόμα πλεονέκτημα των παιχνιδιών είναι το έμφυτο «σύστημα» μέτρησης της απόδοσής του, η οποία μπορεί να είναι είτε μία επιβράβευση με μία νίκη ή ήττα, όπως συμβαίνει στο σκάκι, είτε με ένα σύνολο πόντων, όπως είναι στα διάφορα παιχνίδια τράπουλας. Το συμπαγές πλαίσιο του προβλήματος που υπάρχει στα περισσότερα παιχνίδια αυτού του είδους παρέχει τη δυνατότητα στους ερευνητές να δοκιμάζουν ιδέες και να αξιολογούν εύκολα τα αποτελέσματά τους. Εκτός από αυτό, τα διάφορα παιχνίδια αποτελούν ένα πιο «φιλικό» και «ευχάριστο» περιβάλλον εργασίας για τον ερευνητή, δεδομένου ότι πρόκειται να αφιερώσει ένα σημαντικό χρόνο εργασίας και μελέτης. Η παρούσα εργασία θα ασχοληθεί με το πρόβλημα της δημιουργίας και υλοποίησης ενός πράκτορα, ο οποίος μπορεί να παίζει ικανοποιητικά ένα αιτιοκρατικό παίγνιο ατελούς πληροφόρησης, το παιχνίδι της Ναυμαχίας.

1.2 Συνεισφορά διπλωματικής εργασίας

Αν και η ανάλυση παιχνιδιών, όπως η Ναυμαχία, είναι λιγότερη πολύπλοκη, σε σχέση με τα στοχαστικά παιχνίδια, και έχουν γίνει φιλότιμες προσπάθειες τέτοιας ανάλυσης πάνω στο παιχνίδι που εξετάζουμε με διάφορες τεχνικές που θα αναφέρουμε στη συνέχεια, εμείς θα προσπαθήσουμε να δώσουμε την δική μας πινελιά στο παιχνίδι αυτό με ένα σετ κανόνων “brute force”. Οι κανόνες αυτοί δεν είναι απαραίτητα καινούριοι, ούτε άγνωστοι στο ευρύ κοινό. Συνδυάσαμε εμπειρικά ό,τι γνωρίζαμε για το παιχνίδι, δημιουργήσαμε μερικές νέες στρατηγικές, και χρησιμοποιήσαμε ό,τι υπήρχε ήδη για το παιχνίδι της Ναυμαχίας. Τελικά, δημιουργήσαμε 8 στρατηγικές με τις οποίες μπορεί κάποιος να παίξει το παιχνίδι της Ναυμαχίας και την παραλλαγή του, όχι όμως με την ίδια απόδοση. Οι αποδόσεις των 8 στρατηγικών είναι κλιμακούμενες, καθεμία επόμενη είναι καλύτερη και αναβαθμισμένη από την προηγούμενη. Επίσης, δημιουργήσαμε και μία ξεχωριστή στρατηγική για την «αντίπαλη πλευρά», δηλαδή την πλευρά της τοποθέτησης πλοίων. Τα αποτελέσματα ήταν ικανοποιητικά, καθώς η καλύτερη μας στρατηγική κατάφερε να λήξει το παιχνίδι στις 44 μόλις βολές κατά μέσο όρο για 10x10 board. Στα αποτελέσματα δείχνουμε τους πειραματισμούς που πραγματοποιήσαμε και για την «αντίπαλη πλευρά», αλλά και για την παραλλαγή με το 20x20 board.

1.3 Δομή διπλωματικής εργασίας

Για την εύκολη ανάγνωση της παρούσας διπλωματικής εργασίας, χωρίσαμε το κείμενο σε κεφάλαια και ενότητες.

Στο παρόν κεφάλαιο αναφέρουμε κάποια εισαγωγικά στοιχεία, όπως τον τομέα τον οποίο αφορά η εργασία, ορίζουμε το πρόβλημά μας, και αναφέρουμε την συνεισφορά και τα αποτελέσματά της.

Στο δεύτερο κεφάλαιο μιλάμε γενικά για την ιστορία των παιχνιδιών και της Τεχνητής Νοημοσύνης, τι επιτεύγματα είχανε ανά τα χρόνια σε μερικά από τα πιο γνωστά παιχνίδια, όπως είναι το σκάκι και η ντάμα, και τέλος κάνουμε αναφορά στις διάφορες κατηγορίες των παιχνιδιών, αναλόγως το περιβάλλον στο οποίο παίζονται.

Στο τρίτο κεφάλαιο αναλύουμε το πρόβλημα που έχουμε να λύσουμε, και περιγράφουμε το παιχνίδι με τους κανόνες και τις παραλλαγές του. Επίσης, γίνεται αναφορά στις διάφορες παρόμοιες εργασίες που έχουν ήδη δημιουργηθεί για το συγκεκριμένο παιχνίδι και βλέπουμε τις δικές τους προσεγγίσεις στο πρόβλημα επίλυσης της Ναυμαχίας.

Το τέταρτο κεφάλαιο αφορά την δική μας προσέγγιση στο πρόβλημα που ορίσαμε, με την ανάλυση όλων των στρατηγικών που σκεφτήκαμε για την πλευρά που ψάχνει να βυθίσει τα πλοία, αλλά και μίας στρατηγικής για την «αντίπαλη πλευρά», την πλευρά δηλαδή που είναι υπεύθυνη για την τοποθέτηση των πλοίων.

Στο πέμπτο κεφάλαιο παρουσιάζουμε τα αποτελέσματα των στρατηγικών που δημιουργήσαμε, για κάθε μία περίπτωση που εξετάσαμε.

Τέλος, στο έκτο κεφάλαιο καταγράφουμε τα δικά μας συμπεράσματα από τα αποτελέσματα που είχαμε, τα σχολιάζουμε, και αναφέρουμε τι θα μπορούσε κάποιος να προσθέσει στο μέλλον, μαζί με το τι μάθαμε εμείς προσωπικά από την εργασία αυτή.

2 Ιστορικό υπόβαθρο

Από τις αρχές της δεκαετίας του 1950, και λίγο μετά την κατασκευή του πρώτου υπολογιστή, άρχισαν οι προσπάθειες σχεδιασμού υπολογιστών, οι οποίοι είχαν την ικανότητα να παίζουν τα δημοφιλή παιχνίδια checkers (ντάμα) και σκάκι. Η έρευνα τις επόμενες δεκαετίες εστίασε στο σκάκι, με την κύρια πρόκληση στον τομέα της Τεχνητής Νοημοσύνης να είναι η κατασκευή ενός υπολογιστικού συστήματος, το οποίο θα ήταν σε θέση να ανταγωνιστεί δυνατούς ανθρώπινους αντιπάλους. Ένας άνθρωπος που παίζει σκάκι βασίζεται σε τεχνικές αναζήτησης brute force, προκειμένου να είναι σε θέση να βρίσκει τη βέλτιστη ή την πιο αποδοτική κίνηση σε κάθε στάδιο του παιχνιδιού. Οι Grand Masters στο σκάκι με αυτόν τον τρόπο έχουν φτάσει στο σημείο να κάνουν αυτή την αναζήτηση πολύ γρήγορα. Γι' αυτόν τον λόγο, η σχετική έρευνα επικεντρώθηκε στην ταχύτερη ανάπτυξη των δομών δεδομένων και στην αναζήτηση αλγορίθμων για την περαιτέρω βελτίωση των επιδόσεων των συγκεκριμένων τεχνικών.

Στην εξέλιξη της Τεχνητής Νοημοσύνης συνέβαλε πολύ και η ανάπτυξη της Θεωρίας Παιγνίων, από τα τέλη της δεκαετίας του 1980 έως τα μέσα της δεκαετίας του 1990, η οποία έχει ως σκοπό την εύρεση της στρατηγικής εκείνης που θα είναι σε θέση να φέρει το καλύτερο δυνατό αποτέλεσμα. Παράλληλα, οι νέες τεχνικές και οι διάφορες καινοτόμες ιδέες, καθώς και η σημαντική αύξηση των υπολογιστικών πόρων, λόγω της εξέλιξης του υλικού (hardware) των υπολογιστών οδήγησαν σε μια σειρά από μεγάλες κατακτήσεις. Το πρώτο σημαντικό επίτευγμα ήταν το γεγονός ότι η ντάμα έγινε το πρώτο παιχνίδι, στο οποίο ο παγκόσμιος πρωταθλητής δεν ήταν άνθρωπος, με το πρόγραμμα Chinook να κερδίζει το παγκόσμιο πρωτάθλημα ανθρώπου εναντίον μηχανών (World Man-Machine Championship). Το επόμενο σχετικό ορόσημο ήταν το σκάκι το 1997, όταν τα περίπου 50 χρόνια επιστημονικής προσπάθειας στο συγκεκριμένο πεδίο απέδωσαν τελικά καρπούς, με το πρόγραμμα Deep Blue της IBM να κερδίζει τον έως τότε παγκόσμιο πρωταθλητή Garry Kasparov.

Η σημασία που δίνεται στο σκάκι, είναι ότι το συγκεκριμένο παιχνίδι ανήκει στην κατηγορία των ντετερμινιστικών παιχνιδιών, δηλαδή είναι ένα παιχνίδι το οποίο δεν επηρεάζεται από τον παράγοντα τύχη και δεν περιέχει στρατηγικές «εξαπάτησης». Επιπλέον, στο σκάκι ο κάθε παίχτης δεν κατέχει πληροφορίες οι οποίες είναι άγνωστες στον αντίπαλο, οπότε η κατάσταση του παιχνιδιού καθορίζεται πλήρως από την ακολουθία κινήσεων που θα επιλέξει ο κάθε παίκτης, κάτι το οποίο χαρακτηρίζει το σκάκι ως ένα παίγνιο τέλειας πληροφόρησης.

Σε αυτό το σημείο θα πρέπει να γίνει αναφορά στην κατηγοριοποίηση των παιχνιδιών αναλόγως το περιβάλλον όπου παίζονται. Διακρίνονται σε ντετερμινιστικά και στοχαστικά, σε τέλειας

και ατελούς πληροφόρησης, σε ακολουθιακά και επεισοδιακά, σε στατικά, δυναμικά, και ημιδυναμικά, σε διακριτά και συνεχή, και τέλος σε μονοπρακτορικά και πολυπρακτορικά. Στα παίγνια τέλειας πληροφόρησης, ο κάθε παίκτης είναι σε θέση να παρατηρήσει ολόκληρη την κατάσταση ενός παιγνίου, με αποτέλεσμα ο καθένας τους να κατέχει τις ίδιες ακριβώς πληροφορίες. Χαρακτηριστικά παραδείγματα τέτοιων παιγνίων είναι το σκάκι και η ντάμα, για το λόγο ότι το ταμπλό και όλα τα πιόνια-πούλια των παικτών είναι ορατά και από τους δύο παίκτες. Σε παίγνια ατελούς πληροφόρησης κάποια στοιχεία τους είναι κρυφά από τους αντίπαλους, με αποτέλεσμα να εισάγεται το στοιχείο της αβεβαιότητας για την κατάσταση του παιχνιδιού. Τα περισσότερα παίγνια καρτών είναι παίγνια ατελούς πληροφόρησης, αφού ο κάθε παίκτης έχει στα χέρια του ένα σύνολο από κάρτες, το οποίο δεν μπορούν να δουν οι αντίπαλοι. Ένα τέτοιο παίγνιο είναι το Poker, σε όλες τις παραλλαγές του, αφού στις περισσότερες από αυτές υπάρχουν κάποια χαρτιά τα οποία είναι κρυφά, αλλά και το παιχνίδι της Ναυμαχίας, που είναι μεν αιτιοκρατικό, είναι όμως και παιχνίδι ατελούς πληροφόρησης.

Σχετικά με τα ντετερμινιστικά (αιτιοκρατικά) και στοχαστικά (μη αιτιοκρατικά) παίγνια, η διαφορά τους έγκειται στους παράγοντες που επηρεάζουν την επόμενη κατάσταση. Σε ένα ντετερμινιστικό παίγνιο κάθε επόμενη κατάσταση καθορίζεται απόλυτα από την παρούσα κατάσταση, καθώς και από την επόμενη κίνηση του κάθε παίκτη. Για παράδειγμα, στο σκάκι η επόμενη κατάσταση του παιγνίου καθορίζεται αποκλειστικά από την παρούσα θέση των πιονιών και την κίνηση ενός από αυτά, όπως επιλέγει να πράξει ο κάθε παίκτης. Αντίθετα, στην κατηγορία των στοχαστικών παιγνίων υπάρχουν διάφορα τυχαία γεγονότα τα οποία επηρεάζουν την επόμενη κατάσταση. Για παράδειγμα, το μοίρασμα των φύλλων σε ένα παιχνίδι καρτών ή το ρίξιμο ενός ζαριού είναι στοχαστικά γεγονότα, τα οποία καθορίζουν την επόμενη κατάσταση του παιγνίου και βρίσκονται έξω από τον άμεσο έλεγχο των παικτών.

Η ανάλυση των ντετερμινιστικών παιγνίων είναι μια διαδικασία λιγότερο πολύπλοκη σε σχέση με την ανάλυση των στοχαστικών παιγνίων, και αυτό οφείλεται στην παράμετρο της αβεβαιότητας η οποία εισάγεται από διάφορα τυχαία γεγονότα, τα οποία είναι σε θέση να περιπλέξουν το παιχνίδι και τη διαδικασία λήψης αποφάσεων. Παράλληλα, η λήψη των ορθών αποφάσεων είναι τυπικά μια πιο εύκολη διαδικασία σε ένα παίγνιο τέλειας πληροφόρησης σε σχέση με ένα παιχνίδι ατελούς πληροφόρησης, όπου η απόφαση είναι πολλές φορές αποτέλεσμα πεποίθησης. Αυτό έχει ως αποτέλεσμα, το σκάκι να είναι πιο “εύκολο” παιχνίδι στη διαδικασία λήψης αποφάσεων, αφού είναι ένα ντετερμινιστικό παίγνιο τέλειας πληροφόρησης, σε σχέση με ένα παίγνιο όπως το Poker, όπου υπάρχει άγνοια για την τρέχουσα κατάσταση.

Δεδομένου πλέον ότι οι ηλεκτρονικοί παίκτες είναι σε θέση να κερδίζουν τους καλύτερους ανθρώπινους παίκτες στο σκάκι, η πρόκληση που αντιμετώπιζε η Τεχνητή Νοημοσύνη για δεκαετίες, θεωρείται πλέον ότι έχει ωριμάσει, με αποτέλεσμα πλέον ο τομέας της Τεχνητής Νοημοσύνης να εστιάζει σε πιο σύνθετα παίγνια. Το τάβλι, το οποίο είναι ένα στοχαστικό παιχνίδι τέλειας πληροφόρησης, αποτέλεσε το επόμενο βήμα σε αυτή την προσπάθεια, με τα αποτελέσματα να είναι θεαματικά. Το 1998 ο Richard S. Sutton (Sutton, R.S., Barto, A.G., 1998) διατύπωσε τη μέθοδο TD(λ) για τη μάθηση χρονικών διαφορών (Temporal Difference Learning). Η πιο χαρακτηριστική επιτυχία της μεθόδου TD(λ) είναι το TD–Gammon, το οποίο αφορά το παιχνίδι πόρτες στο τάβλι, που υλοποιήθηκε από τον Tesauro (Tesauro, G., 1995). Με χρήση τεχνικών ενισχυτικής μάθησης (Reinforcement Learning) και ύστερα από 1,5 εκατομμύρια παιχνίδια με τον εαυτό του, το TD-Gammon κατάφερε όχι μόνο να έχει φτάσει σε επίπεδο ανάλογο των πρωταθλητών στο συγκεκριμένο παιχνίδι, αλλά και να εισάγει νέες στρατηγικές στο παιχνίδι κατά το ξεκίνημά του. Ανάλογη επιτυχία έχει επιτευχθεί και στα αιτιοκρατικά παίγνια ατελούς πληροφόρησης, τα οποία συχνά μπορούν να προσεγγιστούν επιτυχώς με μεθόδους brute force. Η κύρια πρόκληση στον τομέα του Θεωρίας Παιγνίων πλέον είναι η πιο περίπλοκη κατηγορία των στοχαστικών παιχνιδιών ατελούς πληροφορίας, όπως είναι για παράδειγμα το Poker.

3 Περιγραφή του παιχνιδιού και καθορισμός του προβλήματος

Το παιχνίδι της Ναυμαχίας, στην αρχική του μορφή με χαρτί και μολύβι, παίζεται σε 4 πλέγματα, δύο για κάθε παίκτη. Τα πλέγματα αυτά είναι τετράγωνα - συνήθως διαστάσεων 10x10 - και κάθε τετραγωνάκι αναπαριστά μία θέση στο πλέγμα, η οποία έχει συντεταγμένες ένα γράμμα και ένα νούμερο. Στο ένα πλέγμα ο κάθε παίκτης τοποθετεί τα πλοία του και σημειώνει τις βολές του άλλου παίκτη. Στο δεύτερο πλέγμα, ο κάθε παίκτης σημειώνει τις δικές του βολές.

Πριν ξεκινήσει το παιχνίδι, κάθε παίκτης τοποθετεί κρυφά τα πλοία του στο κυρίως πλέγμα. Κάθε πλοίο καταλαμβάνει έναν συγκεκριμένο αριθμό συνεχόμενων θέσεων, συνήθως σε ευθεία γραμμή. Μπορεί να τα τοποθετήσει είτε οριζόντια, είτε κάθετα, αλλά όχι διαγώνια. Ο αριθμός των θέσεων κάθε πλοίου είναι ανάλογος του είδους του πλοίου. Τα πλοία δεν μπορούν να τέμνουν το ένα το άλλο, αλλά μπορούν να εφάπτονται. Το πλήθος και οι τύποι των πλοίων είναι πανομοιότυπα για κάθε παίκτη.

Κάθε παίκτης έχει 5 πλοία στη διάθεσή του. Τα πλοία αυτά, με βάση τους κανόνες της έκδοσης του παιχνιδιού του Milton Bradley το 1990, είναι τα εξής:

No.	Τύπος πλοίου	Μέγεθος
1	Aircraft Carrier	5
2	Battleship	4
3	Cruiser	3
4	Submarine	3
5	Destroyer	2

Το 2002, η Hasbro μετονόμασε το Cruiser σε Destroyer, καταλαμβάνοντας πλέον 3 θέσεις, και πρόσθεσε στη θέση του ένα νέο πλοίο δύο θέσεων με όνομα Patrol Boat.

No.	Τύπος πλοίου	Μέγεθος
1	Aircraft Carrier	5
2	Battleship	4
3	Destroyer	3
4	Submarine	3
5	Patrol Boat	2

Στις μεταγενέστερες μορφές του, ως επιτραπέζιο και ηλεκτρονικό παιχνίδι, η Ναυμαχία είναι πιο απλή στο στήσιμο του παιχνιδιού και αποτελείται είτε από δύο διπλά πλαστικά ή ηλεκτρονικά ταμπλό, ένα για κάθε παίκτη. Μόλις οι παίκτες τοποθετήσουν τα πλοία τους, αποφασίζουν τυχαία ποιος θα ξεκινήσει πρώτος και το παιχνίδι αρχίζει με μία σειρά γύρων. Σε κάθε γύρο, οι παίκτες εναλλάξ ανακοινώνουν το τετράγωνο στο οποίο θα ρίξουν τη βολή. Μετά από κάθε βολή ενός παίκτη, είναι υποχρεωτική η δήλωση επιτυχίας (HIT) ή αποτυχίας (MISS) της βολής από τον άλλο παίκτη. Όταν όλα τα τετράγωνα ενός πλοίου έχουν χτυπηθεί (HIT), τότε ανακοινώνεται ότι το συγκεκριμένο πλοίο βυθίστηκε. Ο παίκτης που θα βυθίσει πρώτος όλα τα αντίπαλα πλοία, είναι ο νικητής.

Εικόνα 1 - Board Battleship Game
Πηγή: Skroutz.gr

Βέβαια, παρόλο που το παιχνίδι είναι ένα παιχνίδι δύο ατόμων, μπορούμε να το εξετάσουμε και ως ένα παιχνίδι όπου παίζονται δύο παιχνίδια ξεχωριστά το ένα από το άλλο και παράλληλα, και τελικά να μπορούμε να δούμε πιο προσεκτικά τον έναν από αυτούς τους παίκτες και να τον εξετάσουμε. Το παιχνίδι θα το δούμε όμως και από τις δύο μεριές, αρχικά από την μεριά του παίκτη που προσπαθεί να βρει και να βυθίσει τα πλοία του αντιπάλου, και κατόπιν από την μεριά της «αντίπαλης» πλευράς, δηλαδή του στησίματος των πλοίων.

Το ζητούμενο της εργασίας είναι ο σχεδιασμός και η υλοποίηση ενός πράκτορα με κάποια «ευφυΐα», τέτοια ώστε να μπορούμε να παίξουμε και να ολοκληρώσουμε αποδοτικά το παιχνίδι της Ναυμαχίας, αλλά και κάποιας παραλλαγής του.

Υπάρχουν ποικίλες παραλλαγές στο παιχνίδι της Ναυμαχίας. Στην έκδοση Salvo του 1931, οι παίκτες έχουν την δυνατότητα να επιλέξουν έναν συγκεκριμένο αριθμό θέσεων, και όλες αυτές οι θέσεις βάλλονται ταυτόχρονα. Ένας παίκτης μπορεί αρχικά να στοχεύσει 5 θέσεις ανά γύρο, όσο είναι και το αρχικό πλήθος των διαθέσιμων πλοίων του, και ο αριθμός βολών μειώνεται κάθε φορά που ένα

πλοίο του βυθίζεται. Σε άλλες εκδοχές αυτής της παραλλαγής, ο αριθμός βολών που επιτρέπεται σε κάθε παίκτη είναι είτε σταθερός σε όλο το παιχνίδι, είτε ίσος με το μέγεθος του μεγαλύτερου πλοίου που δεν έχει δεχτεί ακόμα βολή. Ο αντίπαλος παίκτης μπορεί να ανακοινώσει το αποτέλεσμα κάθε βολής ή να αναφέρει απλά πόσα HIT ή MISS υπήρξαν χωρίς αναφορά στη θέση τους. Η συγκεκριμένη παραλλαγή είναι για πιο έμπειρους παίκτες, όπως αναφέρουν οι κανόνες του Milton Bradley.

Άλλες παραλλαγές επιτρέπουν τους παίκτες να μην αναφέρουν ότι ένα πλοίο βυθίστηκε, αναγκάζοντας τους αντιπάλους τους να προβούν σε περισσότερες βολές ώστε να εξασφαλίσουν ότι το πλοίο βυθίστηκε και δεν υπάρχει κάποιο άλλο εκεί κοντά. Μία παραλλαγή επιτρέπει τον παίκτη να μετακινεί ένα πλοίο σε μία καινούργια θέση που δεν έχει χτυπηθεί κάθε 4ο ή 5ο γύρο.

Στην παρούσα εργασία, η παραλλαγή που εξετάσαμε ήταν το μέγεθος του board από 10x10 σε 20x20. Καθώς ο κώδικάς μας είναι πλήρως παραμετροποιήσιμος, θα μπορούμε να εξετάσουμε όποιο μέγεθος board και πλοίων επιθυμούσαμε. Για να μείνει το μέγεθος της εργασίας σε λογικά πλαίσια, είδαμε ενδελεχώς μόνο μία παραλλαγή, την 20x20. Μία ακόμα επέκταση που υλοποιήσαμε με παραμετροποιήσιμο τρόπο είναι η δυνατότητα να ορίσει ο χρήστης στόλους με διαφορετικό αριθμό πλοίων και διαφορετικά μεγέθη, παρόλο που δεν εξετάσαμε ενδελεχώς τέτοια σενάρια.

3.1 Σχετικές εργασίες

Ο Sevenster (2004) εισάγει το πρόβλημα του παιχνιδιού της Ναυμαχίας ως ένα πρόβλημα απόφασης και αποδεικνύει ότι είναι NP-complete. Ένα NP-complete πρόβλημα είναι κάθε πρόβλημα το οποίο ανήκει σε μια κατηγορία υπολογιστικών προβλημάτων για τα οποία δεν έχει βρεθεί ως τώρα ένας αποτελεσματικός αλγόριθμος λύσης. Πολλά σημαντικά προβλήματα της επιστήμης των υπολογιστών ανήκουν σε αυτήν την κατηγορία. Τα λεγόμενα εύκολα ή επιλύσιμα προβλήματα μπορούν να λυθούν με αλγόριθμους υπολογιστών οι οποίοι εκτελούνται σε πολυωνυμικό χρόνο, με την έννοια ότι σε ένα πρόβλημα μεγέθους n, ο χρόνος ή ο αριθμός των βημάτων που χρειάζονται για να βρεθεί η λύση είναι μια πολυωνυμική συνάρτηση του n. Οι αλγόριθμοι για την επίλυση δύσκολων ή δυσεπίλυτων προβλημάτων, από την άλλη πλευρά, απαιτούν χρόνους που είναι εκθετικές συναρτήσεις του μεγέθους n του κάθε προβλήματος. Οι αλγόριθμοι πολυωνυμικού χρόνου θεωρούνται αποτελεσματικοί, ενώ οι αλγόριθμοι εκθετικού χρόνου θεωρούνται αναποτελεσματικοί, επειδή οι χρόνοι εκτέλεσης των τελευταίων αυξάνονται πολύ πιο γρήγορα, καθώς αυξάνεται το μέγεθος του προβλήματος.

Ένα πρόβλημα ονομάζεται NP (μη ντετερμινιστικά πολυωνυμικό) εάν η λύση του μπορεί να βρεθεί και να επαληθευτεί σε πολυωνυμικό χρόνο. Με τον όρο μη ντετερμινιστικά εννοείται ότι δεν ακολουθείται κάποιος συγκεκριμένος κανόνας για να γίνει η εικασία. Εάν ένα πρόβλημα είναι NP και όλα τα άλλα προβλήματα NP ανάγονται σ' αυτό σε πολυωνυμικό χρόνο, τότε το πρόβλημα καλείται NP-πλήρες. Έτσι, η εύρεση ενός αποδοτικού αλγορίθμου για οποιοδήποτε πρόβλημα NP-complete, συνεπάγεται ότι μπορεί να βρεθεί ένας αποτελεσματικός αλγόριθμος για όλα αυτά τα προβλήματα, αφού οποιοδήποτε πρόβλημα ανήκει σε αυτήν την κλάση μπορεί να αναδιατυπωθεί σε οποιοδήποτε άλλο μέλος της κλάσης. Δεν είναι γνωστό αν θα βρεθούν ποτέ αλγόριθμοι πολυωνυμικού χρόνου για προβλήματα NP-complete και ο προσδιορισμός τού αν αυτά τα προβλήματα είναι επιλύσιμα ή δυσεπίλυτα παραμένει ένα από τα πιο σημαντικά ερωτήματα στη θεωρητική επιστήμη των υπολογιστών. Όταν ένα πλήρες πρόβλημα NP πρέπει να λυθεί, μια προσέγγιση είναι η χρήση ενός πολυωνυμικού αλγόριθμου για την προσέγγιση της λύσης. Τέλος, η απάντηση που λαμβάνεται με αυτόν τον τρόπο δεν θα είναι απαραίτητα βέλτιστη, αλλά θα είναι αρκετά κοντά σε αυτήν. Ωστόσο, στην εργασία του ο Sevenster (2004) δεν παρέχει καμία σαφή εικόνα για το πώς ένας πράκτορας που παίζει το παιχνίδι μπορεί να εφαρμοστεί στην παραπάνω περίπτωση.

Οι Bridon, Correll, Dubler, και Gotsch (2009) αναφέρονται στη χρήση ευρετικών αλγορίθμων, δηλαδή αλγορίθμων που παρέχουν αποδεκτές λύσεις, αλλά στερούνται επίσημης απόδειξης, οι οποίες γίνονται ωστόσο όλο και πιο συνηθισμένες στην επίλυση προβλημάτων τέτοιων τύπου.

Χρησιμοποιούνται για την αναζήτηση λύσεων σε προβλήματα NP-complete και είναι ένα δημοφιλές θέμα στην έρευνα της Τεχνητής Νοημοσύνης. Δεδομένου ότι το παιχνίδι της Ναυμαχίας περιλαμβάνει έναν άλλο παίκτη του οποίου η συμπεριφορά είναι άγνωστη, δεν υπάρχει κάποια ευρέως αποδεκτά αλγοριθμική λύση. Ωστόσο, ένας ευρετικός αλγόριθμος μπορεί να βρει μοτίβα που προκύπτουν στο παιχνίδι του αντιπάλου και μπορεί να λύσει ή να κερδίσει ένα παιχνίδι πιο γρήγορα από ό,τι μια αφελής και hard-coded αντιμετώπιση. Άλλοι ερευνητές χρησιμοποιούν μια προσέγγιση που βασίζεται σε γενετικούς αλγορίθμους, οι οποίοι προσαρμόζονται στο στυλ παιχνιδιού του αντιπάλου σε πολλά παιχνίδια. Οι γενετικοί αλγόριθμοι γενικά είναι μια ισχυρή μέθοδος εύρεσης γρήγορων και δυναμικών λύσεων κατά το χρόνο εκτέλεσης (W. Banzhaf, J. R. Koza, C. Ryan, L. Spector and C. Jacob, 2000). Το μειονέκτημα αυτής της μεθόδου είναι το γεγονός ότι το πρόγραμμα πρέπει να ολοκληρώσει μια φάση μάθησης πριν αυτό να μπορεί να παίξει αποτελεσματικά. Οι ίδιοι αναφέρουν ότι στον τομέα της Τεχνητής Νοημοσύνης υπάρχουν τρεις πιθανές προσεγγίσεις που ταιριάζουν καλά σε αυτού του τύπου τα προβλήματα, οι οποίοι είναι τα συστήματα βασισμένα σε κανόνες, οι γενετικοί αλγόριθμοι, και τα νευρωνικά δίκτυα. Και οι τρεις αυτές περιοχές της Τεχνητής Νοημοσύνης μπορούν να εφαρμοστούν αποτελεσματικά στο παιχνίδι της Ναυμαχίας. Τα συστήματα που βασίζονται σε κανόνες χρησιμοποιούν "κανόνες", οι οποίοι δημιουργούνται από τους χρήστες και διέπουν τη συμπεριφορά του αλγορίθμου. Αυτοί οι κανόνες εξαρτώνται από τις γνώσεις του εμπειρογνώμονα που παρέχει ενόραση στο παιχνίδι. Τα συστήματα που βασίζονται σε κανόνες μπορεί να είναι αποτελεσματικά για να επιτύχουν τη βύθιση ενός πλοίου και μπορούν να κωδικοποιηθούν γρήγορα με το σύστημα να βελτιστοποιεί ορισμένα βήματα. Ωστόσο, η έρευνα του Mysinger (1998) έχει αποδείξει ότι ένα τέτοιο σύστημα μπορεί ακόμα να έχει καλύτερη απόδοση από τον κώδικα που δημιουργείται από ένα γενετικό αλγόριθμο. Μια προσέγγιση που χρησιμοποιεί νευρωνικά δίκτυα δεν θεωρείται κατάλληλη από τους Bridon, Correll, Dubler, και Gotsch (2009) λόγω των χαρακτηριστικών του «μαύρου κουτιού» του αλγορίθμου, δηλαδή την αδυναμία να αντιληφθούμε τις εσωτερικές λειτουργίες του συστήματος. Αν και το σύστημα είναι ουσιαστικά πολύ πιο δυναμικό από ένα ειδικό σύστημα βασισμένο σε κανόνες, μια λύση για την τοποθέτηση πλοίων και τη λήψη απόφασης είναι πολύ περίπλοκη σε σύγκριση με μια αυστηρά κωδικοποιημένη στατιστική ανάλυση (Roy, 2000). Σύμφωνα με τους Bridon, Correll, Dubler, και Gotsch (2009), ένα τέτοιο σύστημα στερείται επίσης την ικανότητα να παρέχει εύκολες και γραμμικές λογικές οδηγίες, οι οποίες είναι απαραίτητες για τη βύθιση του πλοίου. Για να είναι ανταγωνιστική μια λύση, πρέπει να μπορεί να εντοπίσει μοτίβα τοποθέτησης πλοίων από τον αντίπαλο, προκειμένου να αυξηθεί η πιθανότητα να χτυπήσει κάποιο εχθρικό πλοίο. Μόλις χτυπηθεί ένα πλοίο, χρειάζεται απλά να εφαρμοστεί μία σχετικά απλή λογική, προκειμένου να βυθίσει τον στόχο. Αν και αυτό θα μπορούσε να είναι hard-coded, υπάρχει απόδειξη

(Mysinger, 1998) ότι οι ευρετικές μέθοδοι μπορεί να είναι πιο αποτελεσματικές από τις συμβατικές μεθόδους.

Σύμφωνα με τους Dellaert, Fox, Burgard, και Thrun (1999), N. Metropolis και S. Ulam (1949), και D. Monniaux (2001), οι μέθοδοι Monte Carlo είναι χρήσιμες όταν εργαζόμαστε με πιθανότητες οι οποίες έχουν μη γραμμικές πυκνότητες πιθανότητας. Συγκεκριμένα, μία μέθοδος εξηγεί πώς να εφαρμόσουμε τις μεθόδους Monte Carlo σε ένα πλέγμα δύο διαστάσεων, ενώ προσπαθεί να εντοπίσει ένα robot σε έναν δεδομένο χάρτη. Αυτό είναι χρήσιμο τόσο για την τοποθέτηση πλοίου, όσο και για τις βολές, καθώς μπορεί να δημιουργηθεί ένας παραλληλισμός μεταξύ εντοπισμού του robot και των πλοίων. Η βύθιση του πλοίου είναι πολύπλοκη, καθώς χρειάζεται λογική, η οποία μπορεί να αλλάξει για να βελτιστοποιήσει τις βολές. Ο γενετικός προγραμματισμός είναι μια έγκυρη μέθοδος αναπαράστασης εντολών και βελτιστοποίησης για γραμμικές λογικές οδηγίες σε ένα τέτοιο πρόβλημα (Benson, 2000).

Στη συνέχεια, στην εργασία του Sahoo, R. (2014), ο συγγραφέας χρησιμοποιεί επίσης Monte Carlo sampling για να δημιουργήσει έναν δυνατό αλγόριθμο για την λύση του παιχνιδιού της Ναυμαχίας. Συγκεκριμένα, ο αλγόριθμος δειγματοληπτεί την belief state των τοποθεσιών των εχθρικών πλοίων N φορές. Έπειτα, επιλέγει να επιτεθεί στην άδεια τοποθεσία, η οποία περιέχει ένα σημείο πλοίου που δεν έχει χτυπηθεί ακόμα στο μεγαλύτερο μέρος των δειγμάτων αυτών και συγκρίνει τα αποτελέσματα του με άλλες υλοποιήσεις. Μεταξύ αυτών, αναφέρεται και στην υλοποίηση των Ahmed, Brown, Colmer, Harton, Doran & Pickford, οι οποίοι παρουσίασαν μια υλοποίηση βασισμένη σε Java, χρησιμοποιώντας έναν πράκτορα αυστηρά βασισμένο σε κανόνες, όπως και η παρούσα εργασία. Αυτοί οι κανόνες είναι hard-coded φωλιασμένες κατασκευές if-else. Ωστόσο, τέτοιες προσεγγίσεις τείνουν να είναι άκαμπτες και ακατάλληλες να αντιδράσουν σε απρόβλεπτες συνθήκες.

Στην εργασία του Clementis, L. (2014), γίνεται μία προσέγγιση του παιχνιδιού μέσω νευρωνικών δικτύων. Αρχικά, σχεδιάζει και περιγράφει ένα heuristic βασισμένο σε πιθανότητες για την στρατηγική που θα ακολουθηθεί για το παιχνίδι. Έπειτα, χρησιμοποιεί subsymbolic feed-forward νευρωνικό δίκτυο με επιτηρούμενη μάθηση και την μέθοδο gradient descent για να λύσει το παιχνίδι της Ναυμαχίας. Στη συνέχεια, θα χρησιμοποιήσει ξανά τον παραπάνω τρόπο, αυτή τη φορά με ενισχυτική μάθηση. Στο τέλος, θα συγκρίνει τους δύο αυτούς τρόπους και την αποδοτικότητα σταδιακής εμφάνισης των στρατηγικών για το παιχνίδι. Καταλήγει ότι το νευρωνικό δίκτυο με επιτηρούμενη μάθηση ήταν το πιο ποιοτικό και αποδοτικό.

Επιπροσθέτως, ο Γουργιώτης, Γ. (2011) στην διπλωματική του εργασία, η οποία υλοποιήθηκε στο Πανεπιστήμιο Πατρών, προσεγγίζει το πρόβλημα επίλυσης ενός παιχνιδιού της Ναυμαχίας με

ευριστικούς κανόνες, δημιουργώντας 4 παίκτες, οι οποίοι σταδιακά γίνονται πιο ευφυείς. Οι παίκτες αυτοί είναι ο τυχαίος, ένας ηλίθιος, ένας έξυπνος, και ένας ιδιοφυής. Ο τυχαίος παίκτης ρίχνει βολές στη τύχη, είτε έκανε μία επιτυχημένη κίνηση, είτε όχι. Ο ηλίθιος συμπεριφέρεται με παρόμοιο τρόπο όπως ο τυχαίος, με τη διαφορά ότι όταν βρει ένα σημείο πλοίου, τότε προσπαθεί να διαλέξει ένα γειτονικό για την επόμενη βολή του. Αν η επόμενη βολή δεν είναι επιτυχία, τότε σύμφωνα με τον συγγραφέα, θα σταματήσει τη προσπάθεια αναζήτησης του πλοίου και θα ξεκινήσει πάλι να ρίχνει τυχαίες βολές. Έπειτα, ο έξυπνος παίκτης θα ψάξει μετά από μία πετυχημένη βολή στη γειτονιά του πλοίου, ώστε να το εντοπίσει και να το βυθίσει, αποφασίζοντας μία κατεύθυνση αρχικά, στα δεξιά, για να ρίξει βολές και λαμβάνοντας υπόψιν κάθε φορά το σημείο όπου βρίσκεται η επιτυχημένη βολή, αλλά και την κατεύθυνση που ψάχνει. Αν η επιτυχημένη βολή είναι στο όριο του board δεξιά, τότε αλλάζει και ψάχνει προς τα αριστερά. Αν ήταν αριστερά, αλλάζει προς τα πάνω, αν ήταν πάνω, αλλάζει προς τα κάτω, και αν ήταν κάτω, τότε παύει να ψάχνει το πλοίο, καθώς το βύθισε ήδη εφόσον έχει ψάξει όλες τις κατευθύνσεις, και συνεχίζει να ρίχνει τυχαίες βολές. Τέλος, ο ιδιοφυής παίκτης δεν ρίχνει τυχαία βολές σε όλο το board, αλλά εφαρμόζει δύο σαρώσεις, ώστε τελικά ο παίκτης να χτυπάει τα σαρωμένα σημεία, που είναι τα μισά σημεία του χάρτη. Στην διπλωματική εργασία του συγγραφέα, δεν έχει υλοποιηθεί κάποιο γραφικό κομμάτι. Ο χάρτης του παιχνιδιού τυπώνεται κάθε φορά στη κονσόλα, αναπαριστώμενος αρχικά με μηδενικά, και στη συνέχεια, αναλόγως τα πλοία και τις βολές, γεμίζει με διάφορα νούμερα.

Τέλος, στην εργασία που έχει αναρτηθεί στην ιστοσελίδα της General Atomics (GA-CCRi), η εταιρεία σύλλεξε στοιχεία από τους εργαζόμενους της, οι οποίοι έπαιζαν μεταξύ τους το παιχνίδι της Ναυμαχίας. Στη συνέχεια, με αυτά τα δεδομένα, δημιούργησαν έναν agent με Deep Reinforcement Learning, χρησιμοποιώντας τη τεχνική προσέγγισης Q-learning, και προπόνησαν τον agent αυτόν με τα δεδομένα αυτά. Τα πειράματα της εταιρείες επικεντρώθηκαν στον τρόπο με τον οποίο θα χρησιμοποιούσαν τα δεδομένα που συλλέξαν για να βελτιώσουν την ικανότητα του agent να παίζει και να νικά απέναντι σε ανθρώπους. Αρχικά, χρησιμοποιήσαν μία απλή Deep Q-learning μέθοδο για να προπονήσουν τον agent, ο όποιος στην αρχή έριχνε το 80% των κινήσεων τυχαία για να εξερευνήσει το board. Στη συνέχεια, η εταιρεία βελτίωσε τον agent, δίνοντας του τα δεδομένα που είχανε συλλέξει από τα παιχνίδια που είχανε παίξει οι εργαζόμενοι της εταιρείας, και με αυτόν τον τρόπο σταδιακά ο agent έγινε αρκετά καλύτερος και έμαθε επιτυχώς πώς να ψάχνει τοποθεσίες και να σημαδεύσει τις θέσεις κοντά σε ένα hit, ώστε να βυθίσει το πλοίο. Ο agent στο τέλος του training είχε μέσο όρο κινήσεων 52, και οι συγγραφείς της εργασίας παρουσίασαν απτές αποδείξεις ότι το μοντέλο τους όντως έμαθε κάποιες στρατηγικές, όπως για παράδειγμα, ο agent συνεχίζει να χτυπά γύρω από

ένα hit και ότι έψαχνε τα πλοία σε διαγώνιες, κάνοντας χρήση του όρου Parity για αποτελεσματικότερη εύρεση πλοίου.

4 Η δική μας προσέγγιση

Στο παρόν κεφάλαιο παρουσιάζεται η δική μας προσέγγιση στο πρόβλημα δημιουργίας ενός αυτόνομου πράκτορα για την παραλλαγή του παιχνιδιού της Ναυμαχίας. Ακολουθήσαμε μία διαφορετική προσέγγιση, βασισμένη σε διάφορους κανόνες και στρατηγικές brute force, οι οποίες σταδιακά γίνονται ολοένα και πιο «έξυπνες».

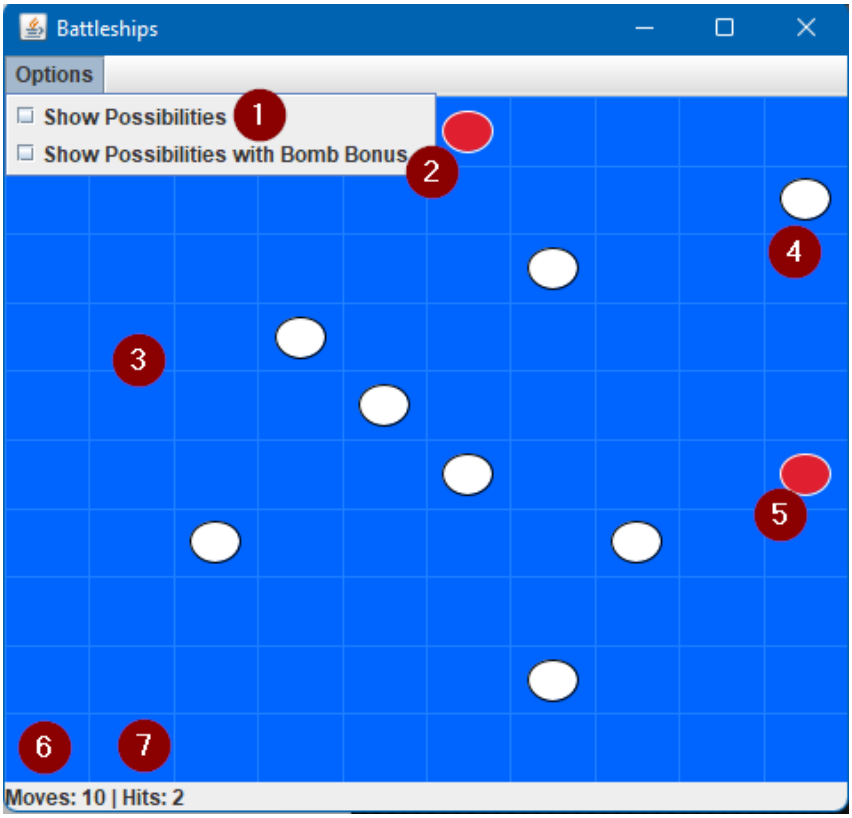
4.1 Υλοποίηση

Το παιχνίδι υλοποιήθηκε από την αρχή σε περιβάλλον Java με 7 κλάσεις, οι οποίες είναι οι εξής:

- **Agent Class**, στην οποία βρίσκεται όλη η λογική των 9 στρατηγικών του πράκτορα μας, χωρισμένες σε συναρτήσεις και if-else statements, αλλά και το πώς συμπεριφέρεται ο πράκτορας γενικότερα.
- **Battleships Class**, η οποία περιέχει τη main συνάρτηση και στην οποία υπάρχει όλη η λογική για τις προσομοιώσεις, η λογική για το command-line όταν τρέχουμε το αρχείο jar, και η λογική να παίξει άνθρωπος το παιχνίδι.
  - Το command-line είναι το εξής: *java -jar Battleships.jar [rows] [columns] [comma separated boat lengths] [player type] [anti-target] [simulation loops]*
    - Οι τιμές στο πεδίο “player type” είναι 0 για human και 1-9 για τις 9 στρατηγικές κατά σειρά.
    - Οι τιμές στο πεδίο “anti-target” είναι 0 ή 1 (true or false).
    - Παράδειγμα: java -jar Battleships.jar 10 10 2,3,3,4,5 9 0 20000
      - Θα γίνουν 20.000 προσομοιώσεις με τη στρατηγική 9 σε ένα board 10x10 με απλή τοποθέτηση με μέγεθος πλοίων 2/3/3/4/5.
- **Board Class**, η οποία δημιουργεί την λογική αναπαράσταση ενός Battleship board, αρχικοποιεί το board, δημιουργεί τυχαία πλοία για τοποθέτηση, και είναι υπεύθυνη για τον υπολογισμό των πιθανοτήτων.
- **BoardListener Class**, η οποία είναι μία κλάση για να ακούει τα board events.
- **Boat Class**, η οποία είναι η λογική αναπαράσταση του πλοίου.
- **Canvas Class**, η οποία είναι υπεύθυνη για την δημιουργία του γραφικού κομματιού (board, βόμβες, πλοία).
- Τέλος, η **Screen Class**, η οποία είναι υπεύθυνη για τη δημιουργία του παραθύρου του παιχνιδιού.

Για το γραφικό κομμάτι, χρησιμοποιήθηκαν οι βιβλιοθήκες AWT και Swing της Java και για το κομμάτι του ήχου που ακούγεται κάθε φορά όταν έχουμε miss ή hit, χρησιμοποιήθηκε η βιβλιοθήκη Sound. Για τις δομές δεδομένων (List – ArrayList – Hashmap κλπ) χρησιμοποιήθηκε εκτενώς η βιβλιοθήκη util.

Παρακάτω δείχνουμε ένα screenshot από το παράθυρο του παιχνιδιού, όπως το βλέπει ο κάθε χρήστης:

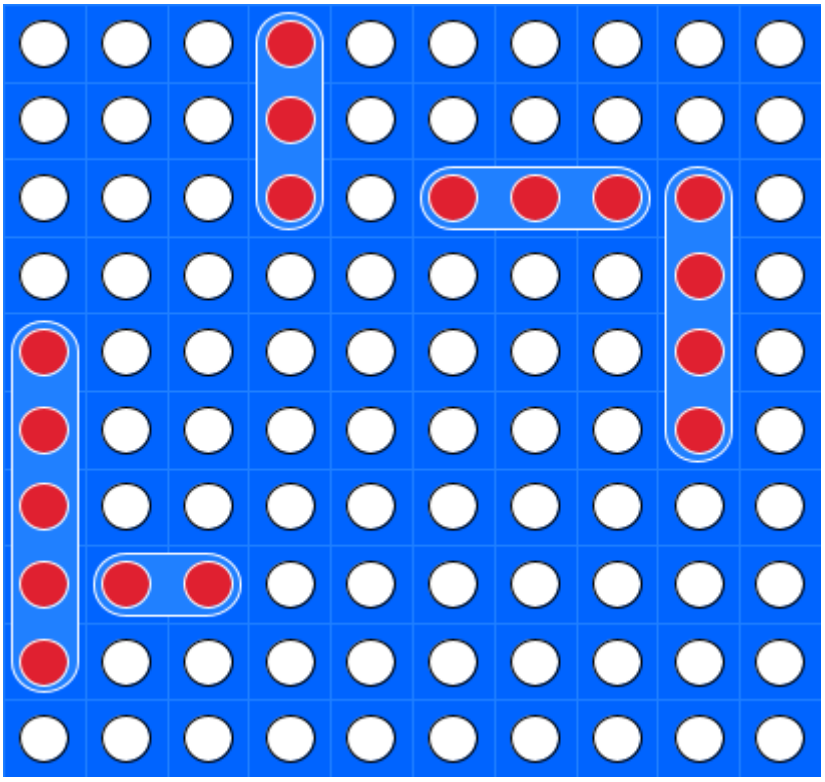


1. Από το μενού “Options”, ο χρήστης έχει τη δυνατότητα να εμφανίσει τις πιθανότητες στο board.
2. Παρομοίως, ο χρήστης έχει τη δυνατότητα να εμφανίσει τις πιθανότητες με το bomb bonus στο board.
3. Το board του παιχνιδιού, στην κλασσική του μορφή 10x10.
4. Ο άσπρος κύκλος δηλώνει αποτυχία, MISS.
5. Ο κόκκινος κύκλος δηλώνει επιτυχία, HIT.
6. Το πλήθος των συνολικών κινήσεων (Moves) που έχει πραγματοποιήσει ο χρήστης κάθε στιγμή.
7. Το πλήθος των επιτυχημένων κινήσεων (Hits) που έχει πραγματοποιήσει ο χρήστης κάθε στιγμή.

Τέλος, υπάρχει η δυνατότητα να τρέξουμε μαζικές προσομοιώσεις, τρέχοντας σε command prompt το command-line με τα κατάλληλα ορίσματα και, αν το επιθυμούμε, η δυνατότητα να αποθηκεύσουμε σε ένα csv αρχείο τις προσομοιώσεις αυτές, γράφοντας μετά το command-line το εξής: > *file_name.csv*.

4.2 Τυχαίος Πράκτορας

Η πρώτη πιθανή στρατηγική για να παίξει κανείς το παιχνίδι με αντίπαλο τον υπολογιστή είναι να ρίχνει βολές στην τύχη. Όπως βλέπουμε παρακάτω, ο τυχαίος πράκτορας αποδίδει κάκιστα, καθώς χρειάστηκε να ρίξει βολές και στις 100 θέσεις του πίνακα.



Σχήμα 4-1: Ολοκληρωμένο παιχνίδι τυχαίου πράκτορα

Τα παιχνίδια χρειάζονται πολλές βολές για να λήξουν, καθώς ο πράκτορας θα επισκεφτεί πολλές θέσεις μέχρι να βρει τα πλοία και να τα βυθίσει. Το σύνολο των θέσεων που εμπεριέχουν πλοία είναι 17, ενώ οι συνολικές θέσεις του board είναι 100. Έτσι, καταλαβαίνουμε τον λόγο για τον οποίο ο τυχαίος πράκτορας αποδίδει τόσο φτωχά. Μάλιστα, το 15% των παιχνιδιών που έτρεξαν σε προσομοιώσεις χρειάστηκαν 100 κινήσεις για να τερματίσουν.

Επιπλέον, η πιθανότητα να έχουμε ένα τέλειο παιχνίδι με την παρούσα στρατηγική υπολογίζεται με τον διωνυμικό συντελεστή και είναι $\frac{n!}{k!(n-k)!}$ όπου n = 100, k = 17, και τελικά το αποτέλεσμα είναι 1 στα 6.650.134.872.937.201.800 παιχνίδια!

Αυτά τα αποτελέσματα φυσικά δεν αρκούν και χρειαζόμαστε μία πιο έξυπνη στρατηγική για το παιχνίδι.

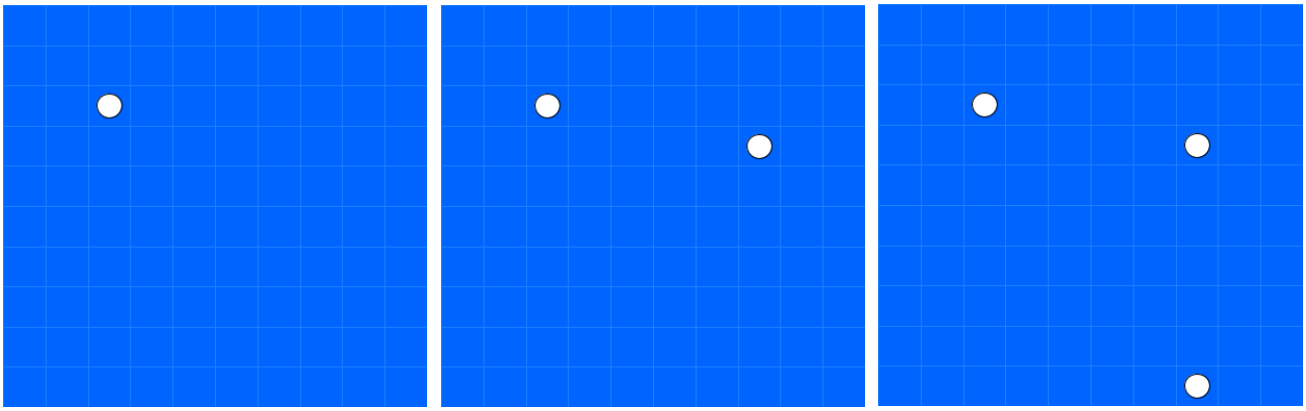
4.3 Δημιουργία έξυπνης στρατηγικής

Δεν ήταν δύσκολο να δημιουργήσουμε μία έξυπνη στρατηγική, η οποία να αποφέρει σημαντικά καλύτερα αποτελέσματα συγκριτικά με τον τυχαίο πράκτορα. Για την υλοποίηση αυτής της στρατηγικής, σκεφτήκαμε τον εξής αλγόριθμο, που τον ονομάσαμε Hunt/Target. Αποτελείται από δύο λειτουργίες, ή modes, το Hunt mode και το Target mode. Αρχικά, στο Hunt mode, ο αλγόριθμος ρίχνει βολές στην τύχη. Όσο οι βολές είναι αποτυχημένες, δηλαδή δεν βρίσκουμε κομμάτι κάποιου πλοίου (MISS), ο αλγόριθμος θα συνεχίζει να ρίχνει τυχαία. Μόλις μία βολή είναι επιτυχημένη (HIT), δηλαδή βρήκαμε κάποιο κομμάτι πλοίου, τότε ενεργοποιείται το Target mode. Στο Target mode, ο αλγόριθμος δημιουργεί μία λίστα όπου αποθηκεύει τους επόμενους πιθανούς στόχους, οι οποίοι μπορεί να βρίσκονται πάνω, κάτω, δεξιά, και αριστερά της επιτυχημένης βολής (HIT). Επιλέγει ένα από τα τέσσερα αυτά σημεία στην τύχη και ρίχνει βολή. Αν η βολή είναι MISS, το αφαιρεί από τη λίστα και συνεχίζει, διαλέγοντας το επόμενο τυχαίο σημείο. Αν η βολή είναι HIT, προσθέτει εκ νέου τα γειτονικά σημεία αυτού του HIT. Ο αλγόριθμος συνεχίζει την διαδικασία αυτή, μέχρι είτε να βυθιστεί πλήρως το πλοίο, είτε να μην υπάρχουν άλλα πιθανά σημεία στη λίστα. Στην περίπτωση αυτή, η λειτουργία Target ολοκληρώνεται, και ο αλγόριθμος επαναφέρει την λειτουργία Hunt και ξεκινά να ψάχνει εκ νέου κάποιο πλοίο.

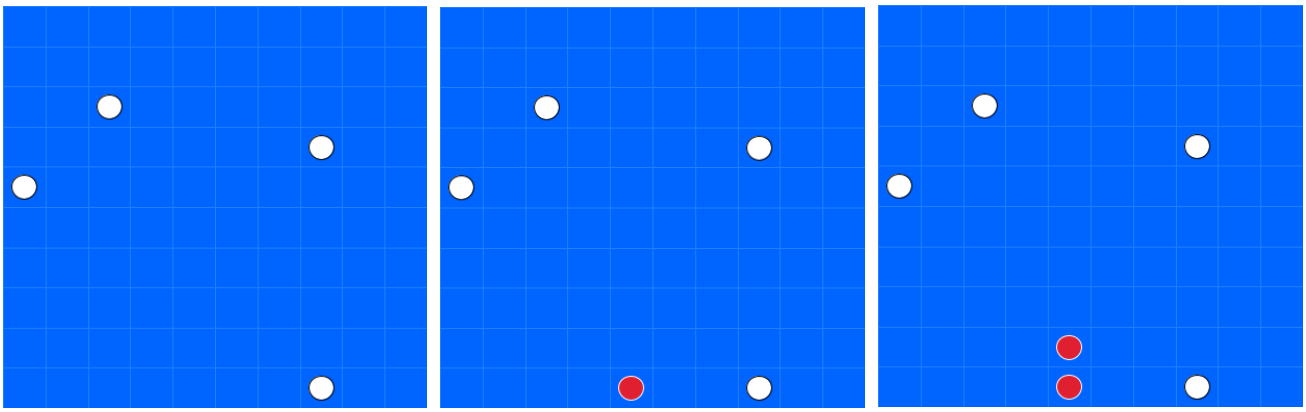
Αν και η συγκεκριμένη στρατηγική έχει εμφανή μειονεκτήματα και δεν είναι επ’ ουδενί τέλεια, είναι μία σημαντική βελτίωση σε σχέση με τον τυχαίο πράκτορα και μας παρήγαγε καλύτερα αποτελέσματα. Βέβαια, δεν είναι αποδοτική, καθώς δεν έχει ιδέα αν ένα πλοίο έχει βυθιστεί, και ο αλγόριθμος ψάχνει όλα τα γειτονικά σημεία όλων των σημείων HIT για να είναι σίγουρος ότι δεν υπάρχει άλλο πλοίο.

4.3.1 Επίδειξη

Παρακάτω ακολουθεί μία επίδειξη της συμπεριφοράς του Hunt/Target αλγόριθμου. Με άσπρους κύκλους αναφερόμαστε στα MISSES και με τους κόκκινους στα HITS.

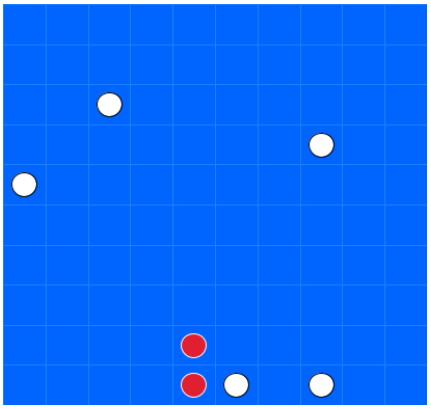


N=1 N=2 N=3

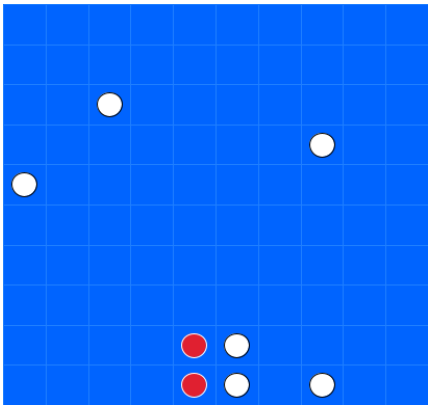


N=4 N=5 N=6

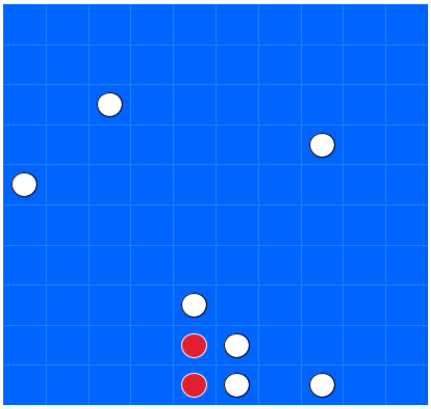
Αρχικά, ο αλγόριθμος βρίσκεται στη λειτουργία Hunt, και ρίχνει τυχαίες βολές στους 4 πρώτους γύρους. Στον γύρο #5, ο αλγόριθμος πέτυχε ένα κομμάτι ενός πλοίου και ενεργοποιεί την λειτουργία Target. Τοποθετεί λοιπόν τα 4 γειτονικά σημεία (στην περίπτωση μας, επειδή το σημείο βρίσκεται στην άκρη του board, τοποθετούνται 3 σημεία, το πάνω, το δεξιό, και το αριστερό κουτάκι) στη λίστα με τους πιθανούς στόχους και επιλέγει ένα στη τύχη στον γύρο #6, το πάνω κουτάκι. Και αυτό είναι HIT, οπότε προσθέτει τα νέα γειτονικά σημεία αυτού του HIT στη λίστα. Στη συνέχεια, επιλέγει τυχαία επόμενα πιθανά σημεία.



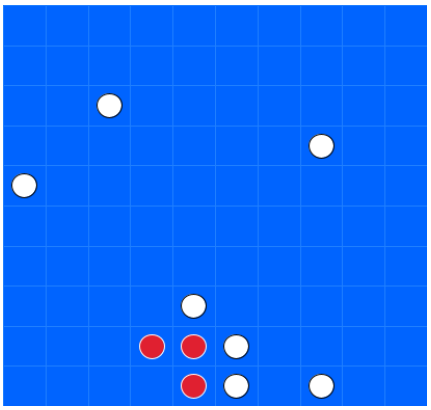
N=7



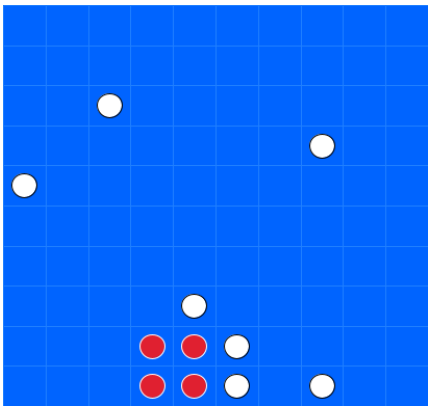
N=8



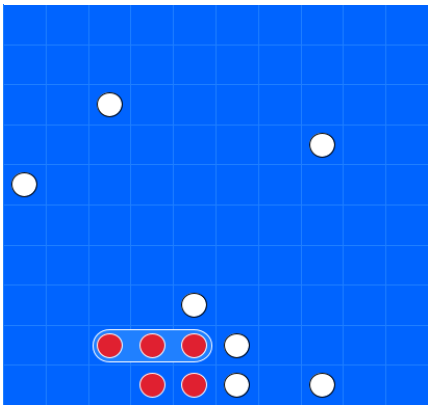
N=9



N=10

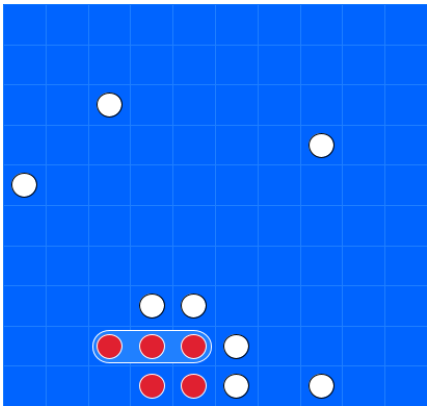


N=11

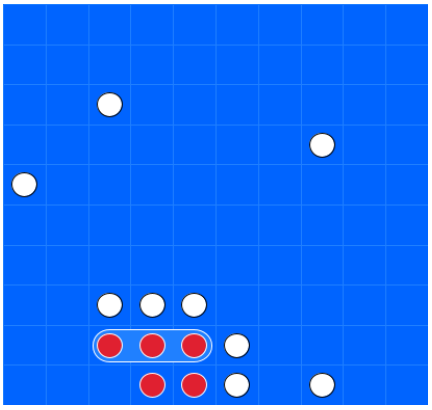


N=12

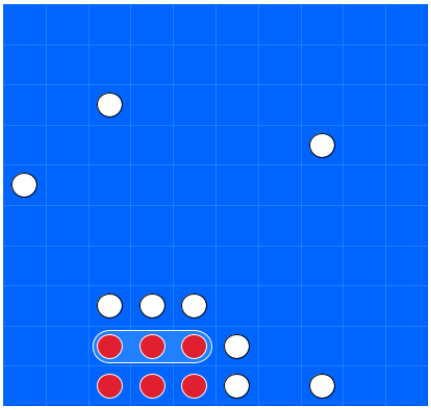
Στους γύρους #7, #8, και #9 είχαμε MISS καθώς δεν υπήρχε κανένα κομμάτι πλοίου. Στον γύρο #10 επιτέλους βρίσκουμε και άλλο HIT, οπότε εκ νέου τα γειτονικά σημεία του εισάγονται στην λίστα. Άλλα δύο επιτυχημένες βολές στους γύρους #11 και #12. Στον γύρο #12, ένα από τα πλοία με μέγεθος 3 βουλιάζει, αλλά ο αλγόριθμος δεν το γνωρίζει, οπότε θα επισκεφτεί όλα τα πιθανά σημεία που βρίσκονται στην λίστα, συμπεριλαμβανομένων και των νέων σημείων που μόλις πρόσθεσε, βουλιάζοντας το πλοίο με μέγεθος 3.



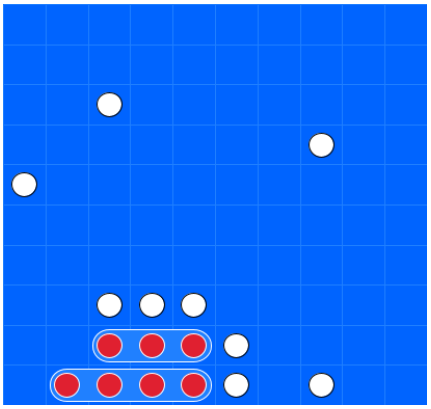
N=13



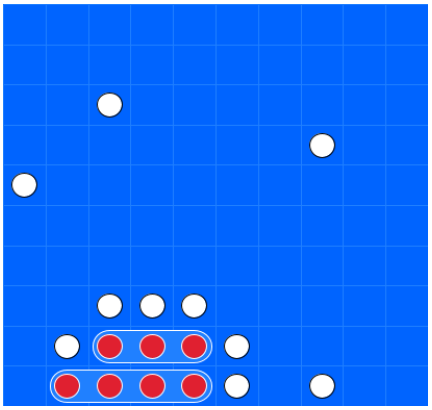
N=14



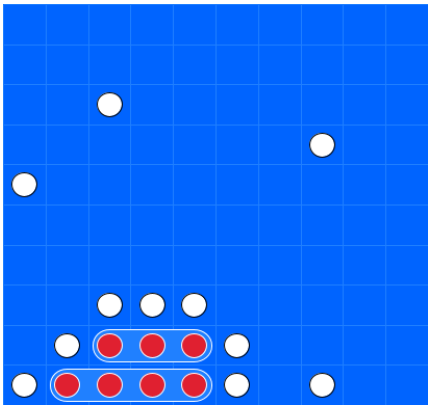
N=15



N=16

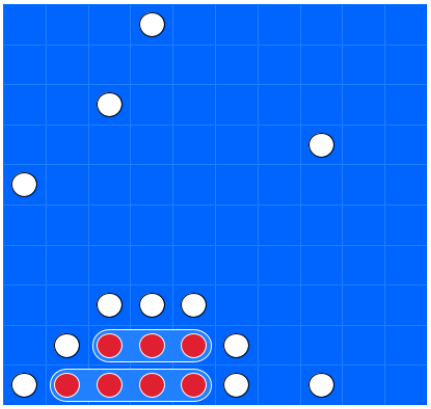


N=17

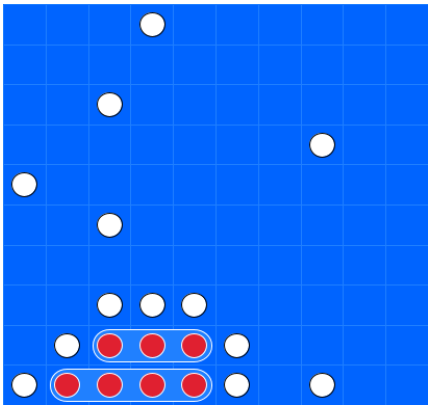


N=18

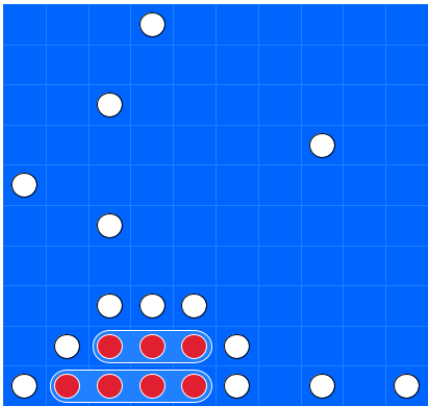
Στον γύρο #16, ο αλγόριθμος βυθίζει και το πλοίο με μέγεθος 4, αλλά καθώς δε το γνωρίζει, εισάγει τα νέα γειτονικά σημεία στην λίστα και θα τα επισκεφτεί όλα. Στον γύρο #18, η λίστα με πιθανούς στόχους είναι πλέον άδεια, και ο αλγόριθμος επαναφέρει τη λειτουργία Hunt στον γύρο #19.



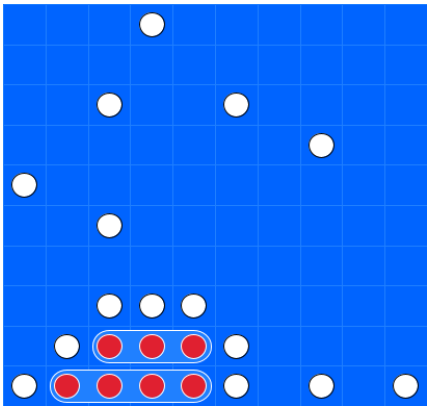
N=19



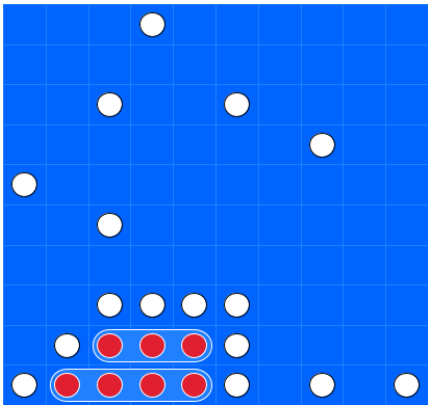
N=20



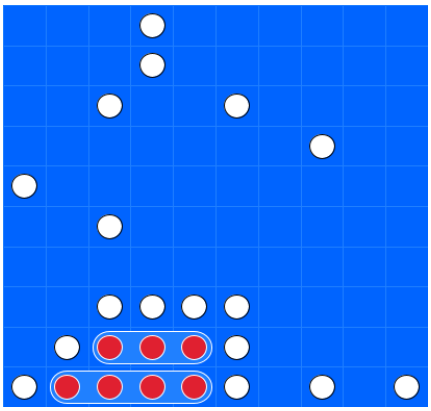
N=21



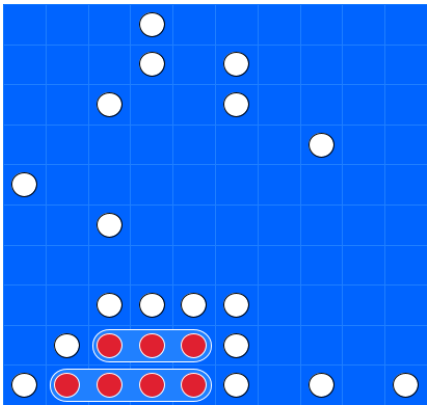
N=22



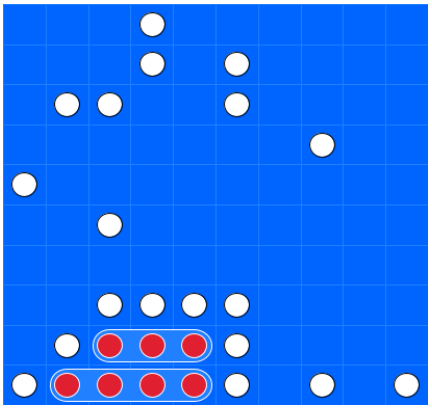
N=23



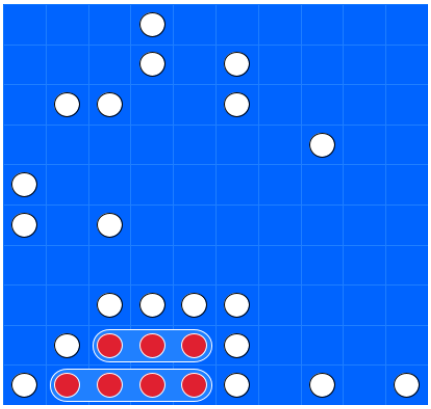
N=24



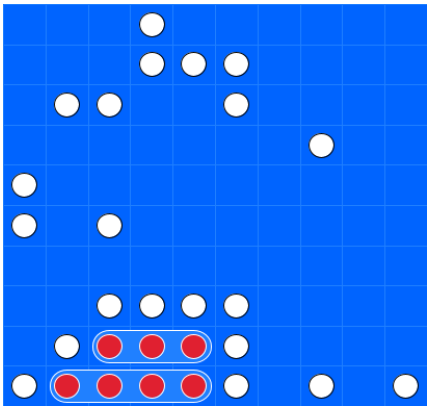
N=25



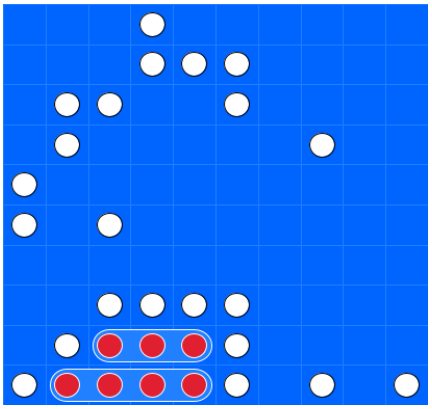
N=26



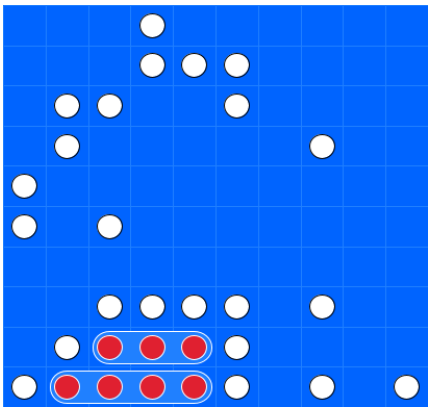
N=27



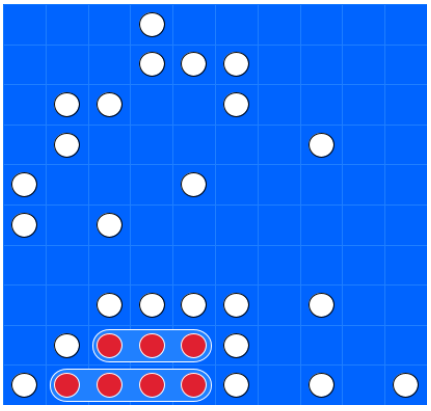
N=28



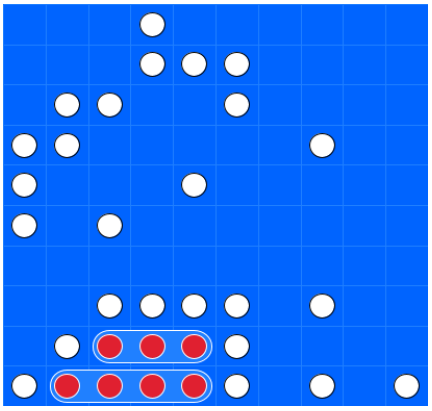
N=29



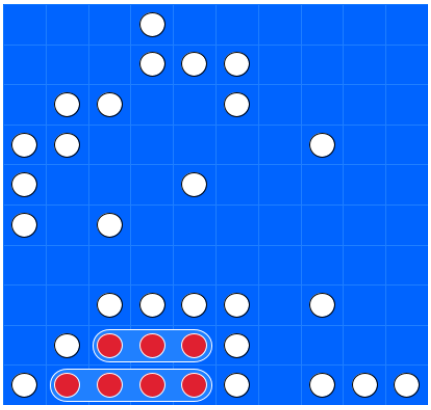
N=30



N=31

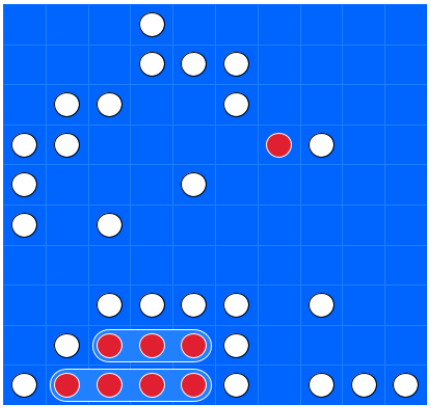


N=32

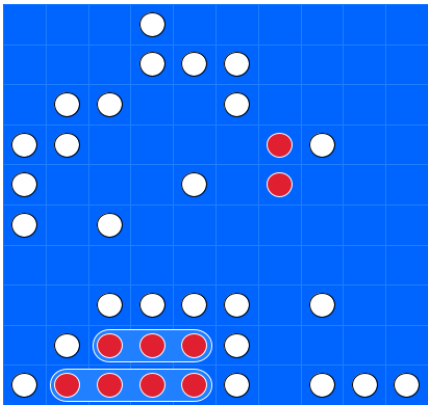


N=33

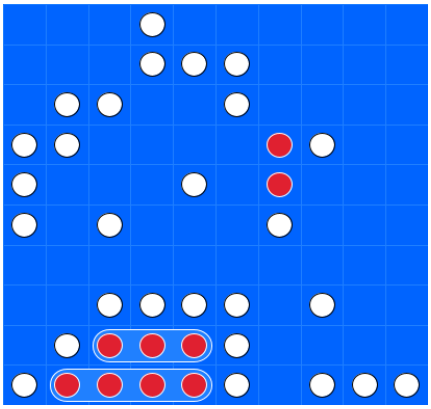
Από τον γύρο #19 μέχρι και τον γύρο #33, ο αλγόριθμος ψάχνει τυχαία να βρει κάποιο πλοίο χωρίς επιτυχία.



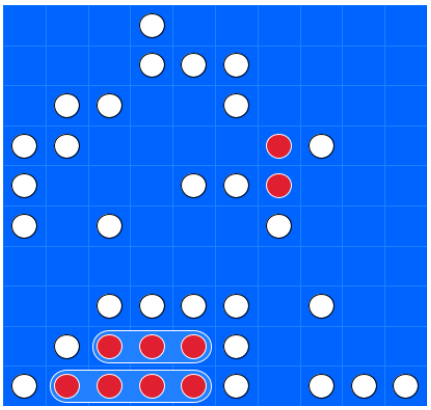
N=34



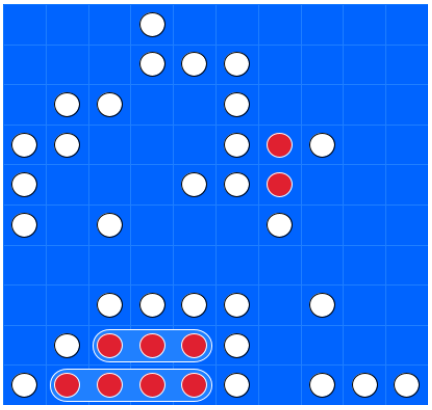
N=35



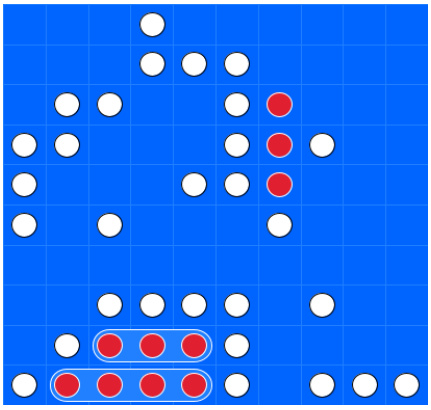
N=36



N=37

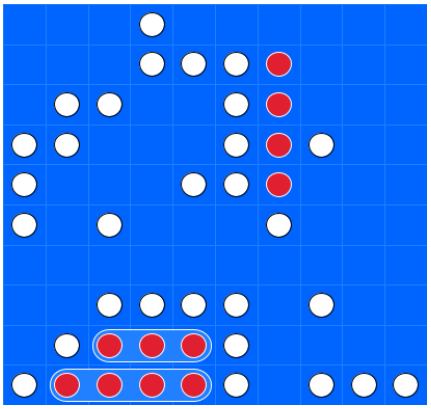


N=38

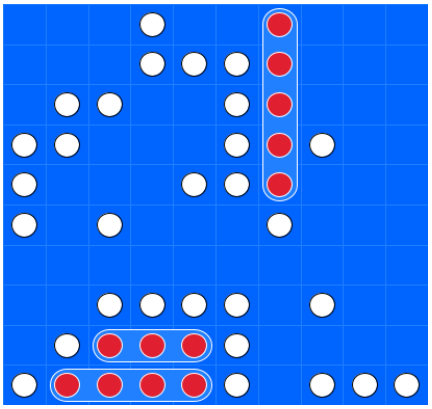


N=39

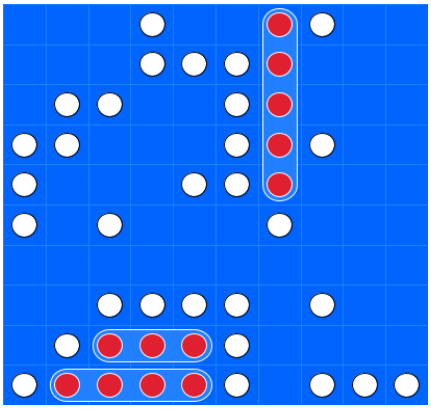
Στον γύρο #34, η τύχη επιτέλους μας ευνοεί και ο αλγόριθμος βρίσκει ένα σημείο πλοίου και ενεργοποιείται η λειτουργία Target ξανά. Επιτυχία και στον γύρο #35, προσθέτοντας φυσικά κάθε φορά όλα τα γειτονικά σημεία, εκτός από όσα έχουμε ήδη επισκεφτεί. Στον γύρο #36 έχουμε MISS, όπως και στους επόμενους δύο γύρους. Στον γύρο #39 βρίσκουμε άλλο ένα σημείο του πλοίου, αλλά δεν έχουμε πολλά πιθανά σημεία που θα εξετάσει ο αλγόριθμος, οπότε και η εύρεση του πλοίου θα γίνει πιο γρήγορα.



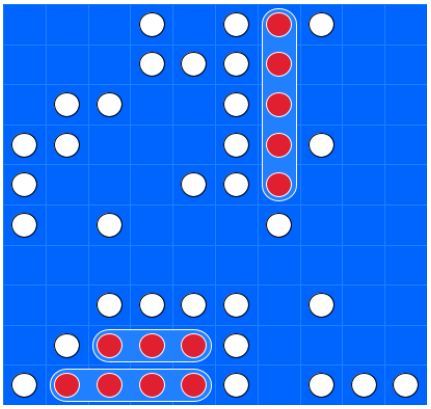
N=40



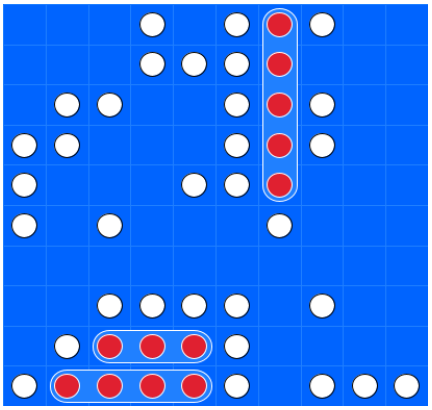
N=41



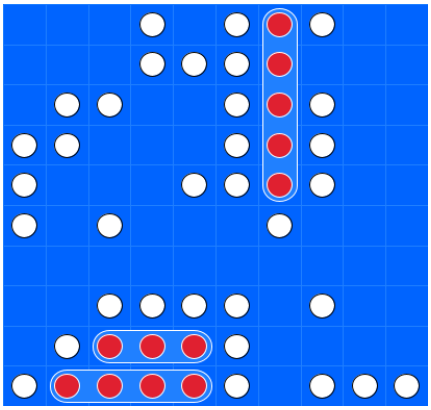
N=42



N=43

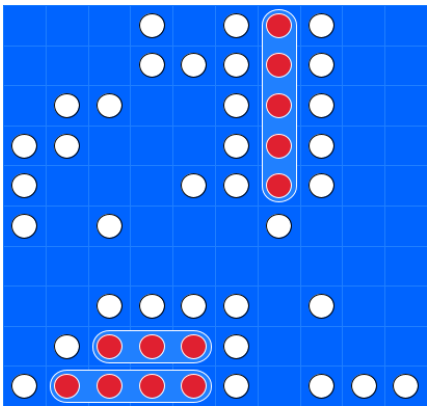


N=44

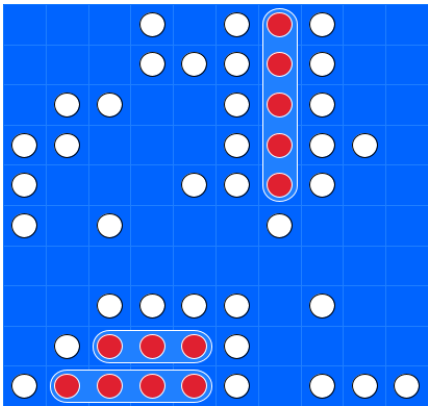


N=45

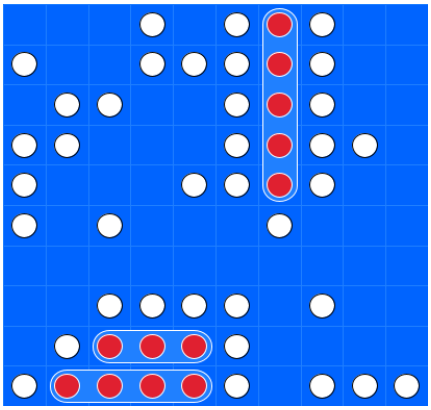
Στους γύρους #40 και #41 ολοκληρώνουμε άμεσα την εύρεση και βύθιση του μεγαλύτερου πλοίου στο παιχνίδι, αλλά καθώς ο αλγόριθμος είναι ακόμα «χαζός», θα εξετάσει όλα τα γειτονικά σημεία, ξοδεύοντας κινήσεις.



N=46

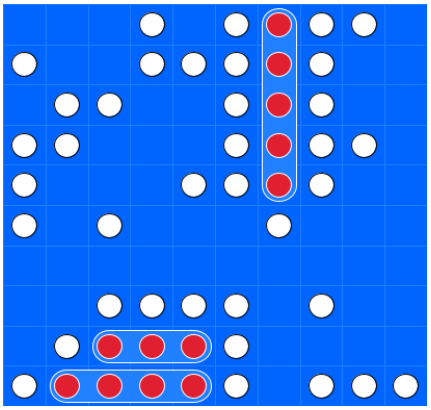


N=47

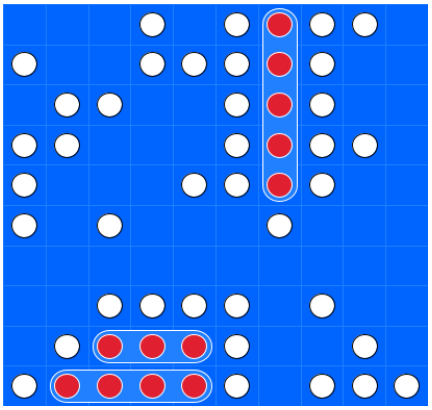


N=48

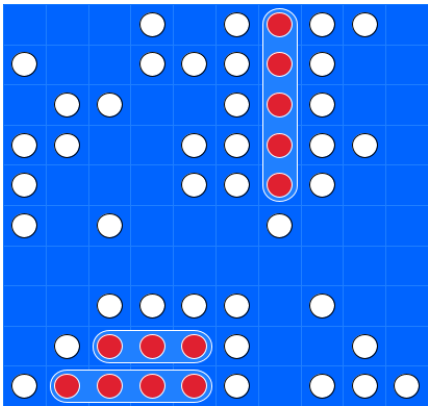
Στον γύρο #46 η λίστα των πιθανών στόχων αδειάζει, και η λειτουργία Hunt είναι ενεργή.



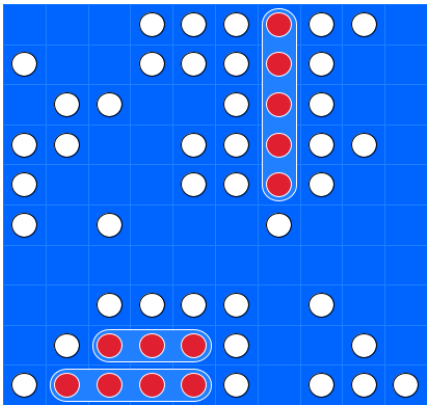
N=49



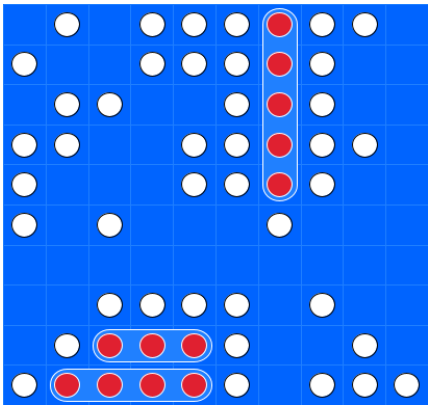
N=50



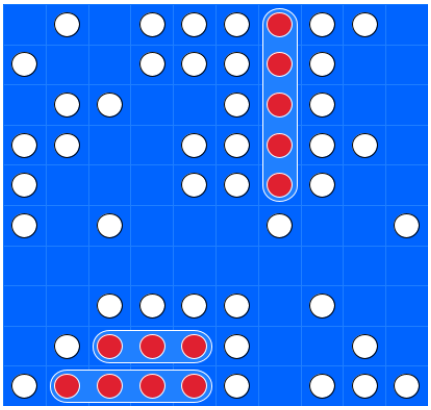
N=51



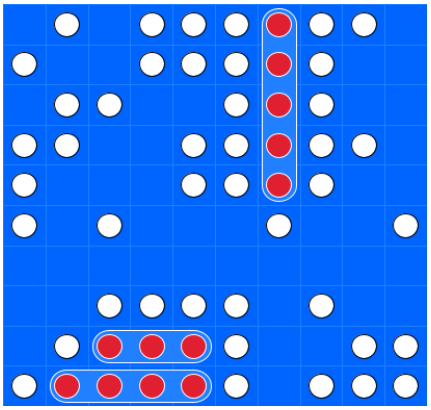
N=52



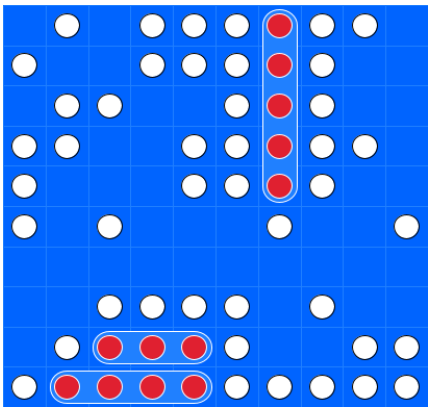
N=53



N=54

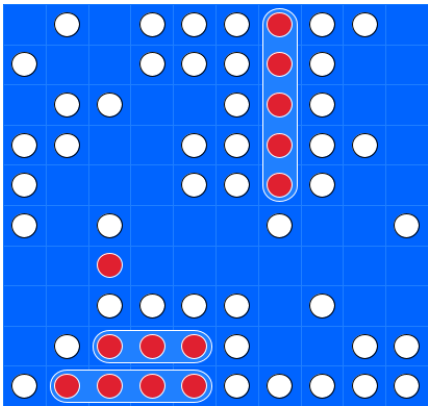


N=55

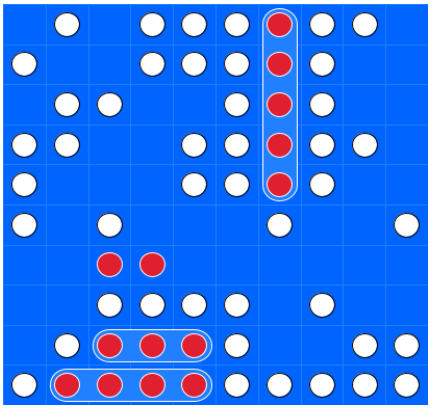


N=56

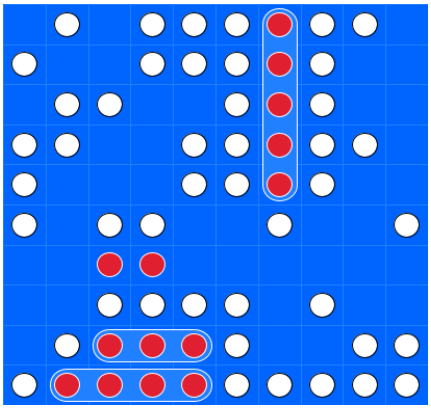
Από τον γύρο #47 μέχρι και τον γύρο #56, δεν υπάρχει καμία επιτυχία και ο αλγόριθμος ψάχνει στα τυφλά.



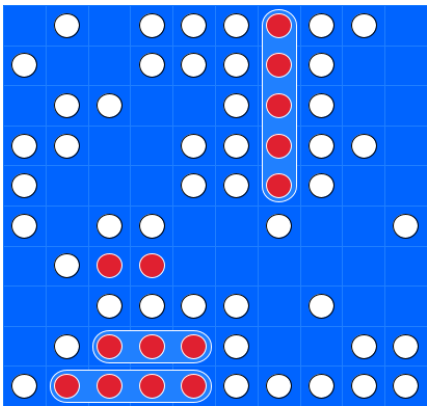
N=57



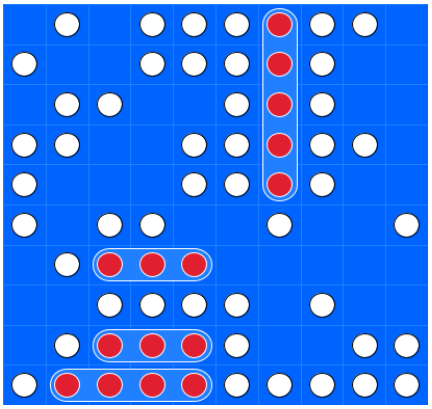
N=58



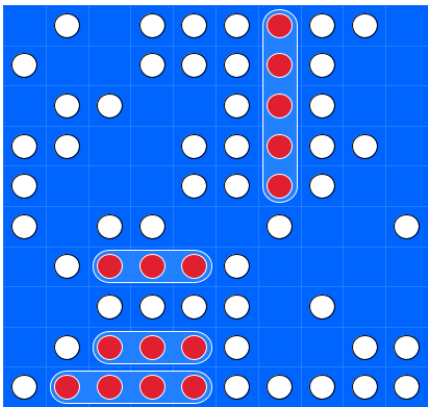
N=59



N=60

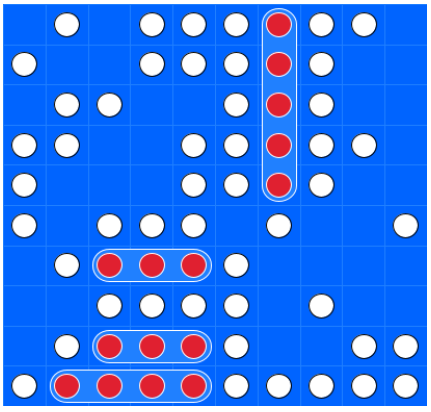


N=61

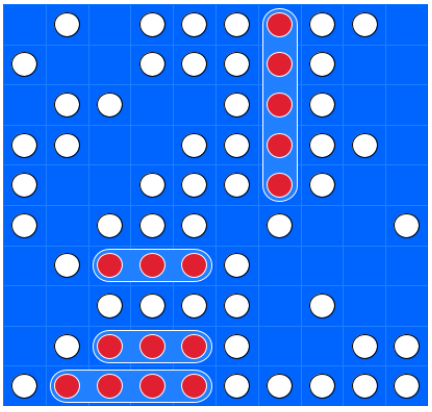


N=62

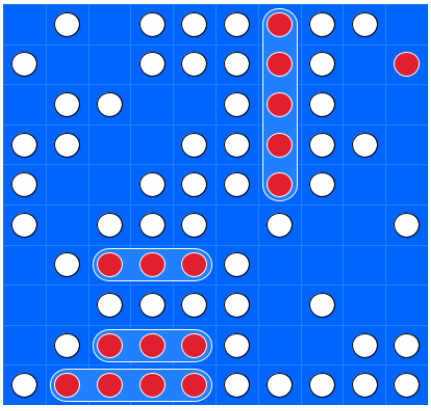
Στον γύρο #57, άλλο ένα πλοίο εντοπίζεται, ενεργοποιώντας την λειτουργία Target. Ξεκινάει να εξερευνεί τα γειτονικά σημεία. Επιτυχία στον γύρο #58, δύο συνεχόμενες αποτυχίες στους γύρους #59 και #60, και η επιτυχία στον γύρο #61 εξασφαλίζει τη βύθιση και του 4^{ου} πλοίου.



N=63



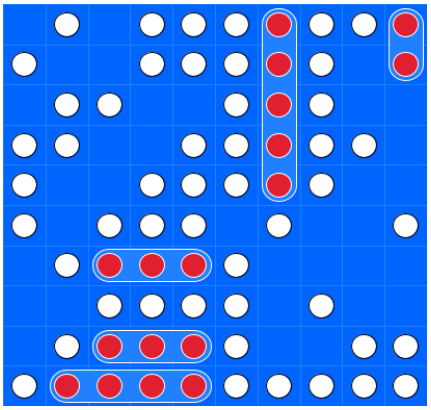
N=64



N=65

Ο αλγόριθμος θα ξοδέψει άλλη μία κίνηση στον γύρο #63 για να επισκεφτεί όλους τους πιθανούς στόχους, η λειτουργία Target απενεργοποιείται, και ενεργοποιείται η λειτουργία Hunt. Έχουμε MISS στον γύρο #64. Στον γύρο #65 στεκόμαστε πολύ τυχεροί και βρίσκουμε το τελευταίο πλοίο, που είναι

και το μικρότερο. Το ίδιο τυχεροί είμαστε και στον γύρο #66, όπου από τα 3 πιθανά σημεία στην λίστα, βρίσκουμε αμέσως το δεύτερο σημείο του πλοίου. Στον αλγόριθμο μας έχουμε βάλει μία συνθήκη τερματισμού, με την οποία όταν ο μετρητής που έχουμε για τα HITS φτάσει τα 17, τα οποία είναι το σύνολο των κουτιών που καταλαμβάνουν τα 5 πλοία μας με μεγέθη 2-3-3-4-5, ο αλγόριθμος τερματίζει. Στην προκειμένη περίπτωση, χρειάστηκαν βολές στις 66 από τις 100 θέσεις για να λήξει το παιχνίδι.



N=66

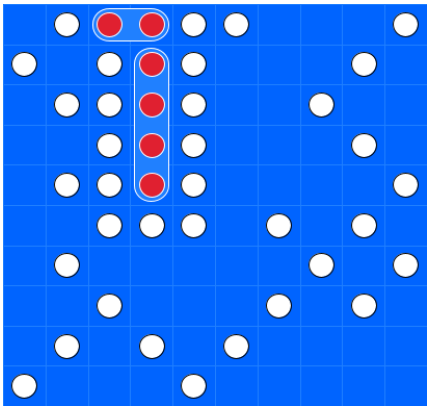
4.4 Βελτίωση έξυπνης στρατηγικής

Αν και ο αλγόριθμος Hunt/Target είναι σημαντικά καλύτερος από την τυχαία προσέγγιση, επιδέχεται περαιτέρω βελτιώσεις. Μία βελτίωση στη λειτουργία Hunt για να μην ψάχνει τυχαία **όλα** τα κουτάκια (στην περίπτωση μας 100, καθώς έχουμε 10x10 board), είναι να ψάξει τα **μισά** κουτάκια. Καθώς το ελάχιστο μέγεθος πλοίου που έχουμε στο παιχνίδι είναι 2, δεν χρειάζεται να ψάχνουμε όλα τα κουτάκια στο board, αλλά μόνο τα κουτάκια με ζυγή ή μονή αρίθμηση. Ακόμα και το μικρότερο πλοίο, με οποιαδήποτε τοποθέτηση και αν έχει, θα καταλαμβάνει αναγκαστικά πάντα ένα ζυγό και ένα μονό κουτάκι. Ο μαθηματικός όρος για να περιγράψει την σκέψη αυτή ονομάζεται Parity.

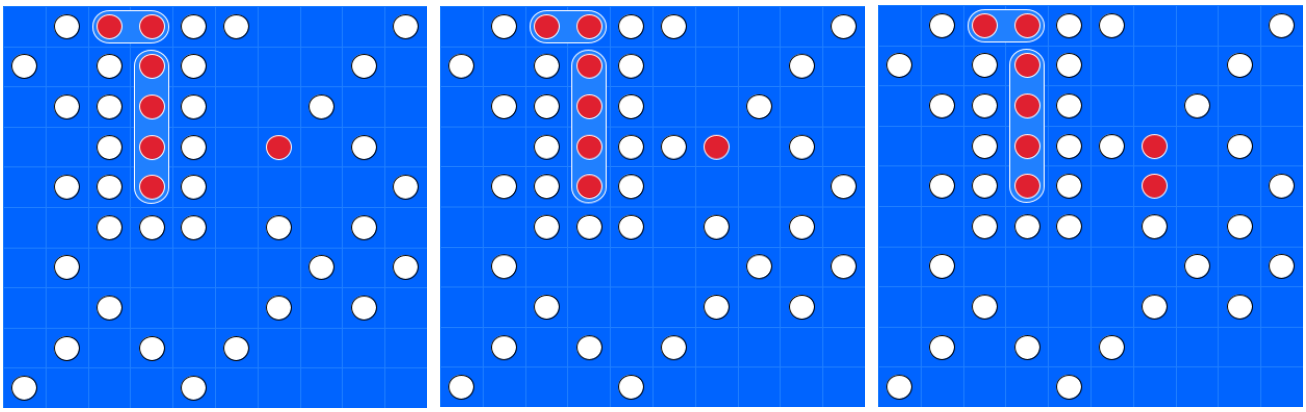
Έτσι, στην αρχή κάθε παιχνιδιού, ορίζουμε αν θα έχουμε ζυγό ή μονό Parity, τυχαία. Στη συνέχεια, ο αλγόριθμος ξεκινά με τη λειτουργία Hunt και αντί να έχει διαθέσιμα 100 κουτάκια, έχει φιλτράρει τα 50 και έχει κρατήσει μόνο τα υπόλοιπα 50. Σε αυτά τα 50 φιλτραρισμένα κουτάκια κρύβεται τουλάχιστον μία θέση από κάθε πλοίο. Μόλις βρει μία θέση από κάποιο πλοίο, τότε ενεργοποιείται η λειτουργία Target, και συνεχίζει την κλασσική συμπεριφορά που δείξαμε προηγουμένως, ασχέτως με τον Parity περιορισμό.

Παρακάτω ακολουθούν μερικά παραδείγματα του αλγόριθμου με τον περιορισμό Parity.

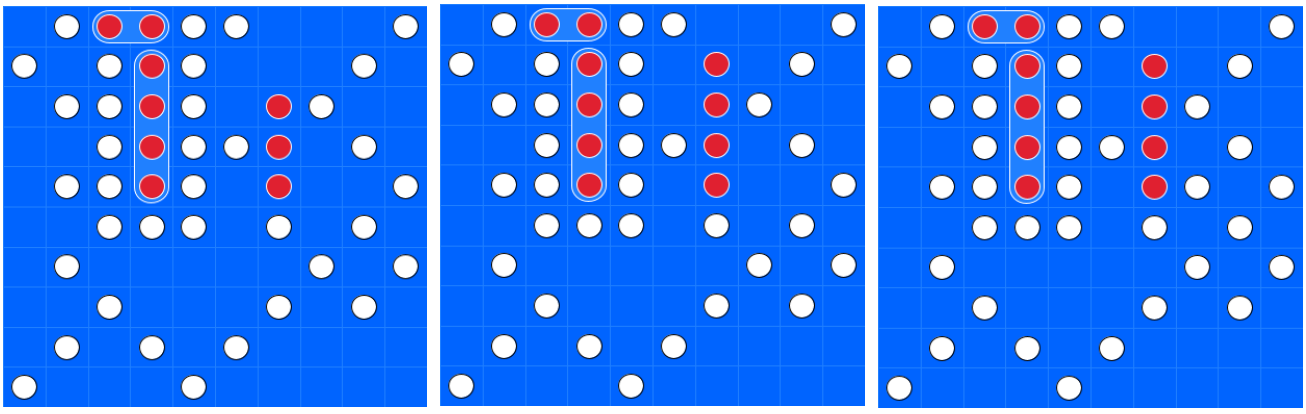
Εδώ βλέπουμε ότι με την προσθήκη του Parity, ο αλγόριθμος φαίνεται να χτυπά σε διαγώνιους,

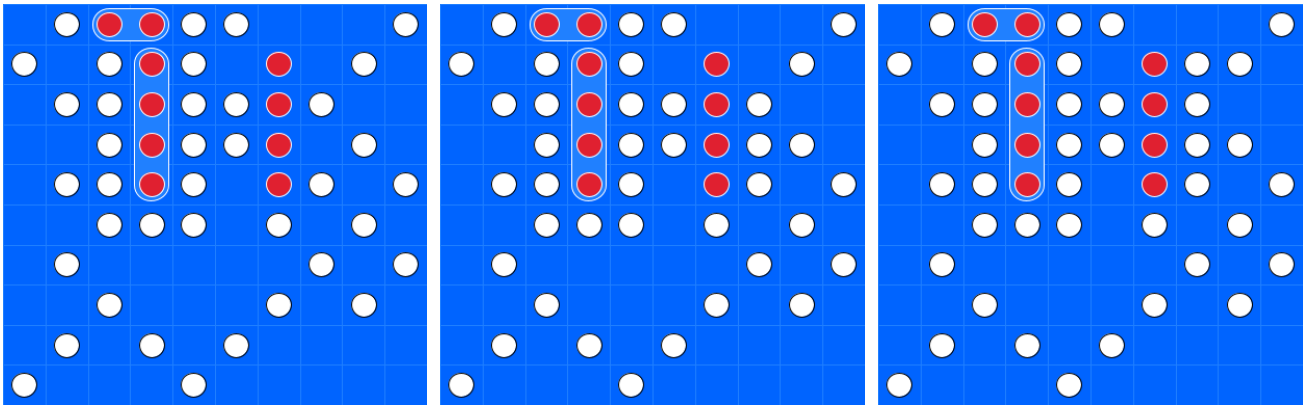


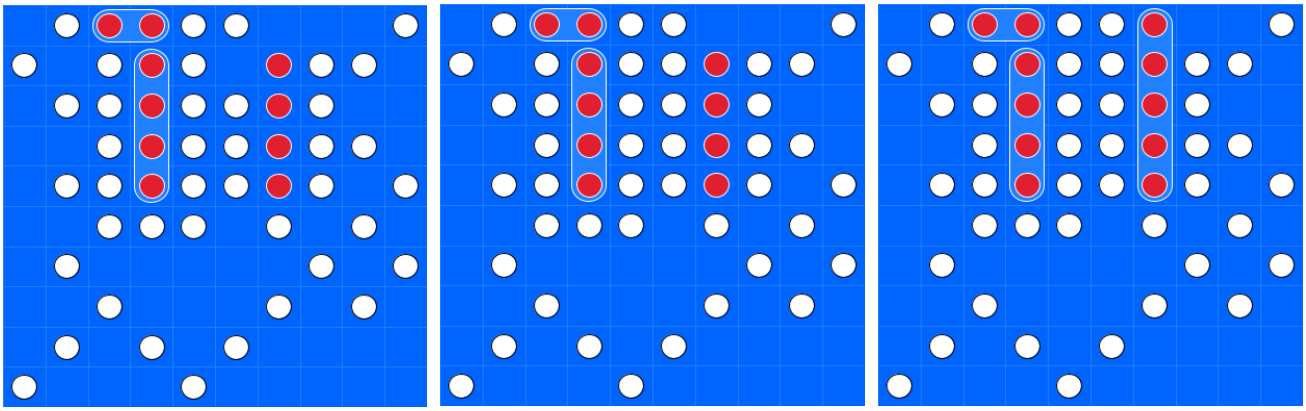
εξερευνώντας όλα τα φιλτραρισμένα κουτάκια.



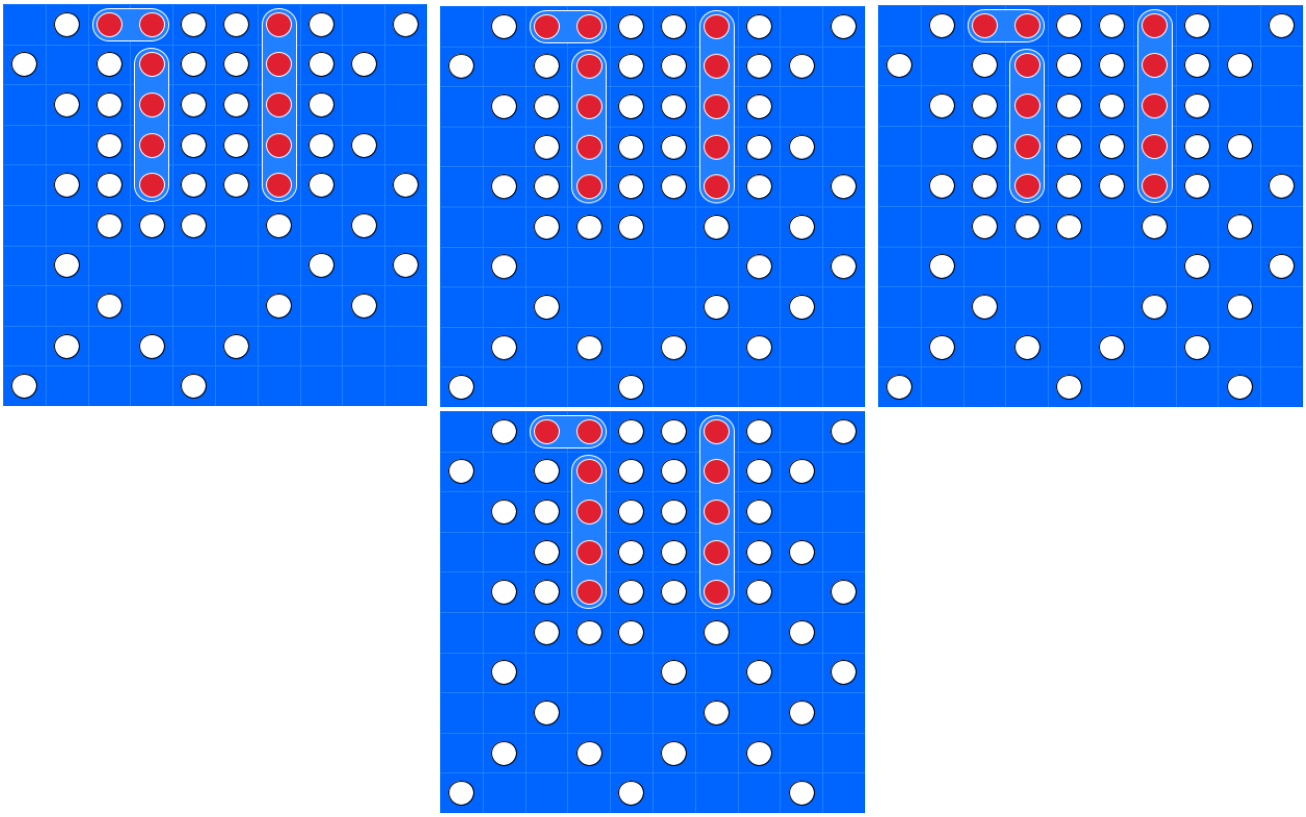
Μόλις όμως βρει HIT, η λειτουργία Target ενεργοποιείται και δεν ασχολούμαστε με τον περιορισμό Parity. Ακολουθείται η κανονική συμπεριφορά της λειτουργίας Target, δηλαδή πρόσθεση γειτονικών σημείων HIT στη λίστα με τους πιθανούς στόχους.







Ο αλγόριθμος ολοκληρώνει τη βύθιση του αεροπλανοφόρου και συνεχίζει ξανά με τα φιλτραρισμένα κουτάκια.

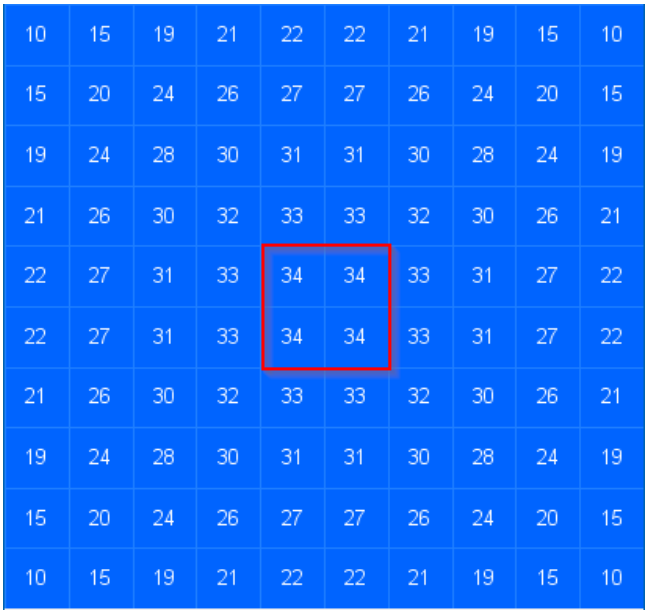


4.5 Εισαγωγή πιθανοτήτων

Η προσθήκη της έννοιας του Parity βοήθησε στη μείωση των συνολικών κινήσεων, αλλά υπάρχει χώρος για περισσότερη βελτίωση στη λειτουργία Hunt, μιας και αυτή είναι και η πρώτη λειτουργία με την οποία ξεκινάει ο αλγόριθμος και η λειτουργία που χρειάζεται άμεση βελτίωση. Η νέα σκέψη είναι ο αλγόριθμός μας να υπολογίζει την πιο πιθανή θέση που μπορεί να βρίσκεται ένα πλοίο για να ρίξει βολή, βάσει ενός υπολογισμού υπέρθεσης όλων των πιθανών θέσεων που κάθε αντίπαλο πλοίο μπορεί να βρίσκεται. Στην αρχή κάθε νέου γύρου, δεδομένου του πλήθους των πλοίων που είναι ακόμα αβύθιστα, ο αλγόριθμος θα υπολογίζει όλες τις πιθανές θέσεις που κάθε πλοίο μπορεί να χωρέσει, οριζόντια ή κάθετα. Αρχικά, αυτό θα ισχύει για όλες τις θέσεις, αλλά καθώς το παιχνίδι θα προχωρά, οι πιθανές θέσεις που θα μπορεί να κρύβεται ένα πλοίο θα μειώνονται. Στη συνέχεια, θα εξηγήσουμε λίγο πιο αναλυτικά πώς ακριβώς υλοποιήσαμε το κομμάτι των πιθανοτήτων.

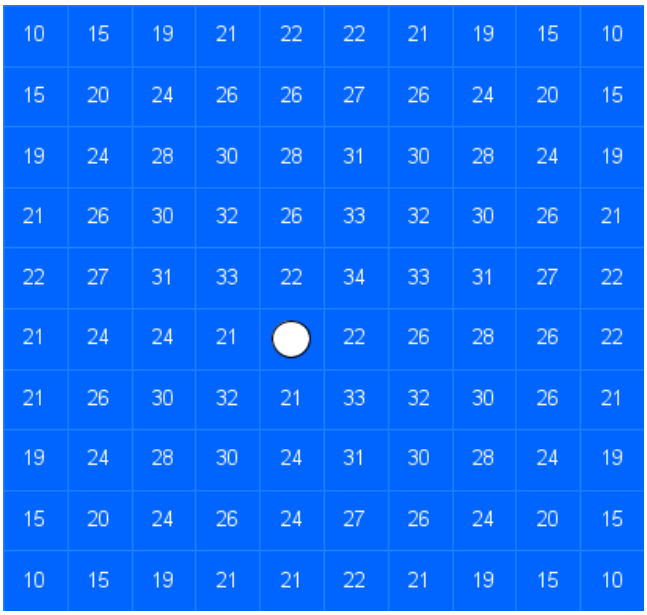
Αρχικά, ας πάρουμε για παράδειγμα το μεγαλύτερο πλοίο στο παιχνίδι, το Aircraft Carrier. Ξεκινώντας από την πάνω αριστερή γωνία του board, προσπαθούμε να το τοποθετήσουμε οριζόντια και να δούμε αν χωράει. Αν χωράει, αυξάνουμε μια μεταβλητή που έχουμε δώσει σε κάθε κελί, τον μετρητή του, κατά 1, για κάθε κελί που το πλοίο καλύπτει. Αυτές οι 5, στην προκειμένη περίπτωση, θέσεις, είναι πιθανές θέσεις που θα μπορούσε να βρίσκεται ένα πλοίο. Έπειτα, μετακινούμε κατά μία θέση δεξιά το πλοίο και επαναλαμβάνουμε την ίδια διαδικασία, μέχρι να φτάσουμε στο τέλος της πρώτης σειράς. Αφού τελειώσει η πρώτη σειρά, μετακινούμε το πλοίο στην δεύτερη σειρά και επαναλαμβάνουμε. Μόλις τελειώσουν όλες οι σειρές, ξανακάνουμε την ίδια διαδικασία, αυτή τη φορά με το πλοίο τοποθετημένο κάθετα. Στον αλγόριθμο μας, κάνουμε την διαδικασία αυτή για κάθε αβύθιστο πλοίο σε κάθε γύρο. Στο τέλος, το board θα έχει τα κελιά του με διαφορετικές τιμές στους μετρητές, αφού δεν χωράνε σε όλα τα κελιά ο ίδιος αριθμός πλοίων.

Παρακάτω βλέπουμε τη μορφή του board στην αρχή του παιχνιδιού. Όπως περιμέναμε, στο κέντρο του board χωράνε περισσότεροι συνδυασμοί πλοίων, είτε οριζόντια, είτε κάθετα, και έτσι βλέπουμε τα 4 κεντρικά κουτάκια με τους μεγαλύτερους μετρητές.

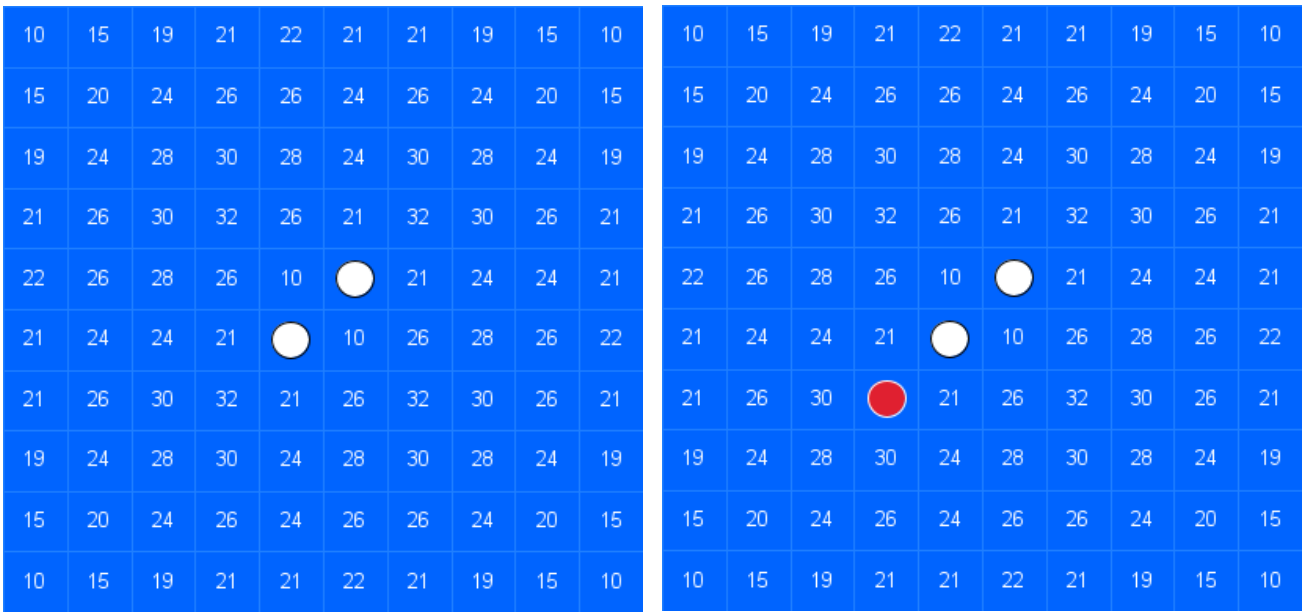


Επίσης, οι 4 γωνίες του board χωρούν τους λιγότερους συνδυασμούς πλοίων, συγκεκριμένα κάθε πλοίο από μία φορά οριζόντια και μία φορά κάθετα. Οι αριθμοί αυτοί είναι ενδεικτικοί της πιθανότητας να υπάρχει πλοίο στην αντίστοιχη θέση.

Όταν ξεκινήσει το παιχνίδι και ρίξουμε την πρώτη βολή και είναι MISS, για παράδειγμα, σε ένα από τα 4 κεντρικά κουτιά, οι πιθανότητες αλλάζουν στην ευρύτερη περιοχή της βολής, καθώς πλέον μειώθηκαν οι πιθανοί συνδυασμοί πλοίων.



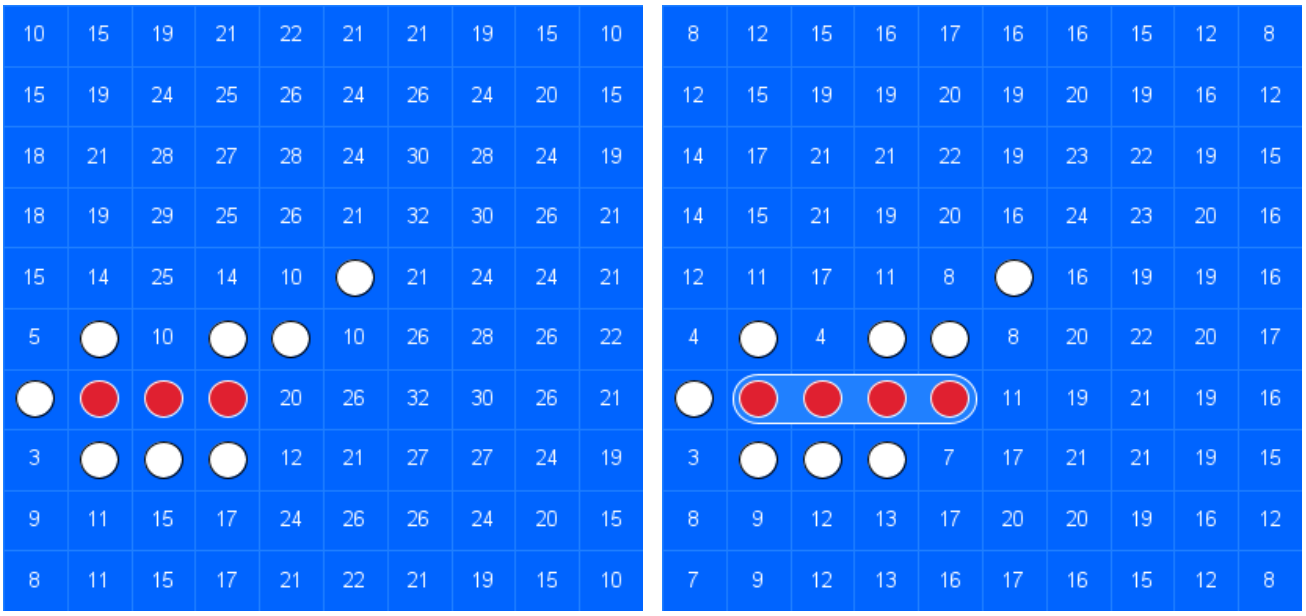
Προφανώς, συνεχίζοντας τη ρίψη βολών, οι πιθανότητες αλλάζουν και διαμορφώνονται ανάλογα.



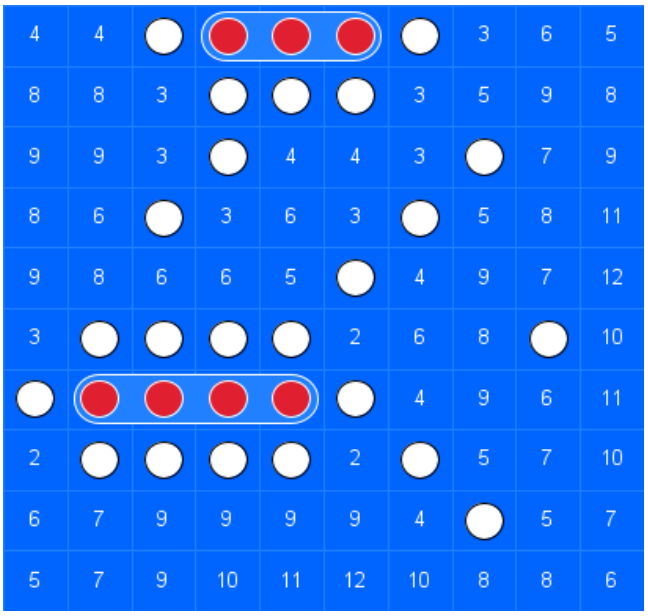
Κάποιες φορές, ένα πλοίο θα χωράει σε κάποιο κελί, και άλλες φορές δε θα χωράει. Όσο το παιχνίδι θα εξελίσσεται, και θα εμφανίζονται hits, misses, και βυθισμένα πλοία, το πλήθος των πιθανών θέσεων για τα πλοία θα μειώνεται.

Η συμπεριφορά του αλγόριθμου είναι η ίδια με πριν όταν βρίσκεται στην λειτουργία Target. Όταν έχουμε HIT, βάζει τα γειτονικά σημεία στη λίστα και τα επισκέπτεται. Μόλις εξαντλήσει όλα τα σημεία, επανέρχεται στη λειτουργία Hunt, η οποία όπως είπαμε δεν χτυπάει τυχαία πλέον, αλλά επισκέπτεται το κουτάκι με τον μεγαλύτερο αριθμό μετρητή (μεγαλύτερη πιθανότητα).

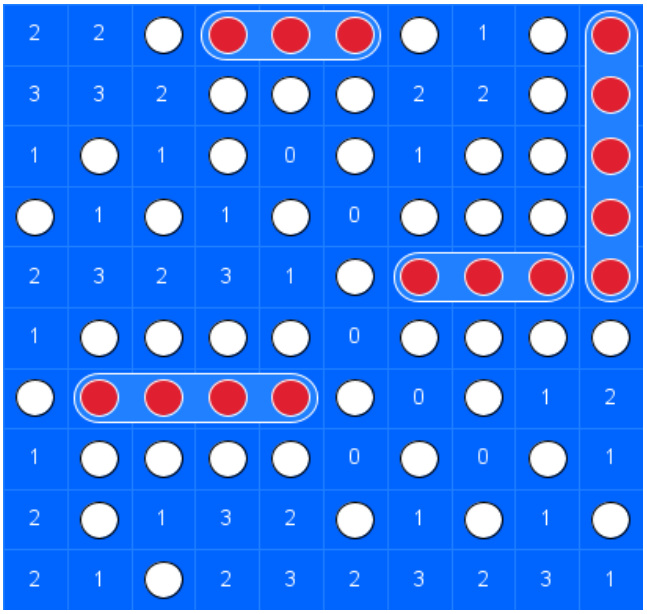
Όταν βυθιστεί ένα πλοίο, οι πιθανότητες σε όλο το board μειώνονται, καθώς πλέον έχουμε 4 πλοία που εξετάζουμε για τις πιθανές τους θέσεις, όχι 5. Επειδή όμως ο αλγόριθμος είναι ακόμα λίγο «χαζούλης», δεν ξέρει ότι βυθίστηκε πλοίο, και θα συνεχίσει να εξετάζει όλα τα πιθανά σημεία στη λίστα μας, ξοδεύοντας πολλές κινήσεις. Στο συγκεκριμένο παράδειγμα, το πλοίο βυθίστηκε στον γύρο #12 και ο αλγόριθμος μας παρέμεινε στην λειτουργία Target μέχρι και τον γύρο #15.

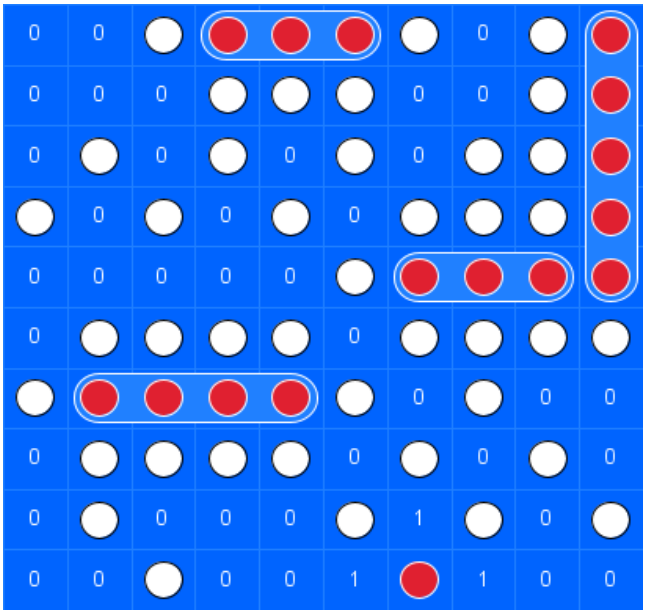


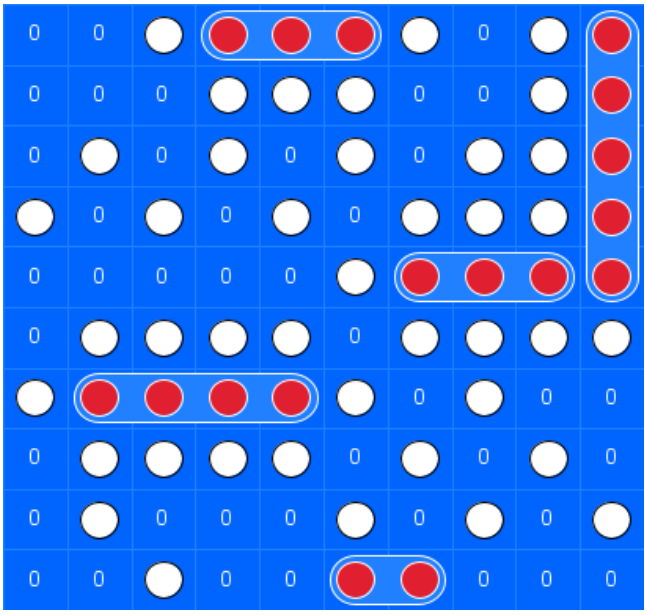
Παρακάτω βλέπουμε το board με δύο βυθισμένα πλοία και πολλά MISS. Βλέπουμε ξεκάθαρα ότι υπάρχουν περιοχές όπου είναι αδύνατον να βρίσκεται το Aircraft Carrier για παράδειγμα.



Τέλος, παρακάτω βλέπουμε το board προς το τέλος του παιχνιδιού, όπου μένει 1 πλοίο αβύθιστο. Πολλά κουτάκια έχουν πλέον μετρητή 0, καθώς το πλοίο που απομένει δε θα μπορούσε να χωρέσει εκεί ποτέ. Με το που βρίσκει ο αλγόριθμος ένα σημείο του πλοίου, αμέσως το board ενημερώνεται και όλα τα υπόλοιπα κουτάκια, εκτός των γειτονικών, μηδενίζονται, καθώς πλέον με δεδομένο ένα HIT, κάποιο πλοίο **πρέπει** να περνάει από το συγκεκριμένο HIT. Το παιχνίδι τελειώνει στον επόμενο γύρο, καθώς βρήκαμε και το πλοίο με μέγεθος 2. Το παιχνίδι ολοκληρώνεται σε 58 γύρους.



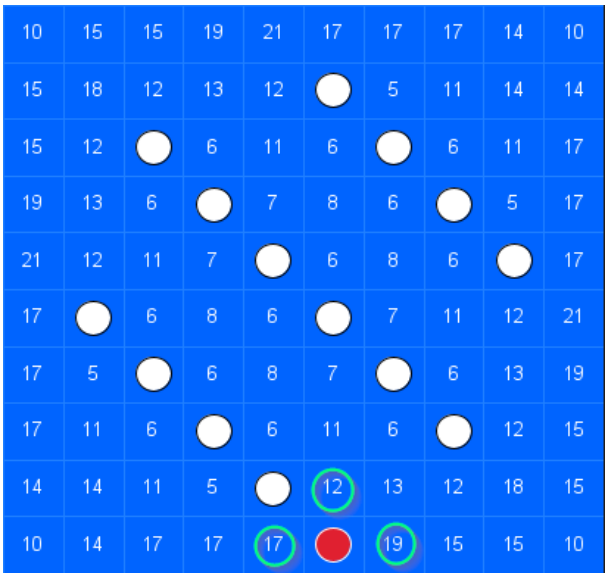




4.6 Στρατηγική “Target Line”

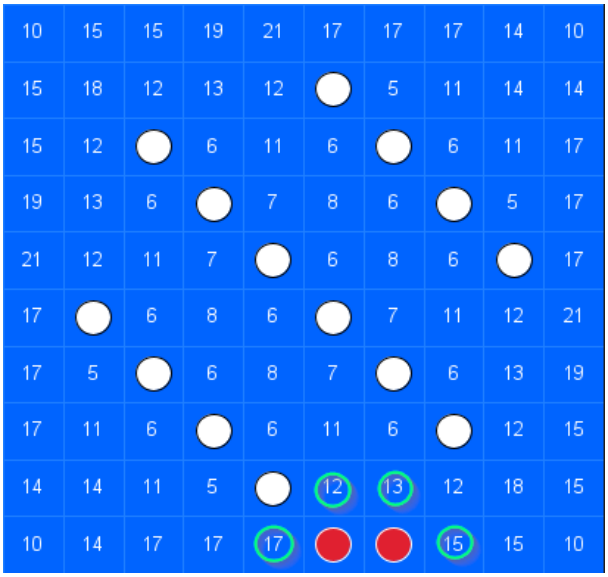
Η βελτίωση με τις πιθανότητες στη λειτουργία Hunt του αλγορίθμου ήταν σημαντική, και θα την υιοθετήσουμε γενικότερα, αλλά το κομμάτι Target ακόμα πάσχει. Μία προσθήκη για να αλλάξουμε λίγο το κομμάτι αυτό είναι η στρατηγική “Target Line”. Με την συγκεκριμένη στρατηγική, θέλουμε ο αλγόριθμος να συνεχίζει να ρίχνει βολές σε ευθεία γραμμή, όπως θα έκανε και κάποιος άνθρωπος αν έπαιζε. Αν και τελικά, όπως θα παρουσιάσουμε αργότερα, τα αποτελέσματα της είναι ελάχιστα καλύτερα συγκριτικά με τον αλγόριθμο χωρίς αυτήν, η χρησιμότητα της φαίνεται αργότερα όταν συνδυαστεί με άλλες στρατηγικές.

Όσον αφορά τη λειτουργία της, είναι αρκετά απλή. Ψάχνουμε στις 4 καρτεσιανές κατευθύνσεις κάθε σημείου που είναι στην λίστα με τους πιθανούς στόχους κατά την λειτουργία Target, για να μετρήσουμε τα ήδη χτυπημένα HIT κουτάκια που μπορεί να υπάρχουν σε κάποια κατεύθυνση. Ψάχνουμε τόσα κουτάκια, όσα είναι και το μήκος του μεγαλύτερου αβύθιστου πλοίου. Στο τέλος, ο αλγόριθμος επιλέγει το σημείο με το μεγαλύτερο σκορ, δηλαδή αυτό που έχει σε κάποια κατεύθυνση περισσότερα HIT κουτάκια από τα υπόλοιπα. Παρακάτω θα δείξουμε ένα παράδειγμα για το πώς ακριβώς συμπεριφέρεται η συγκεκριμένη στρατηγική.



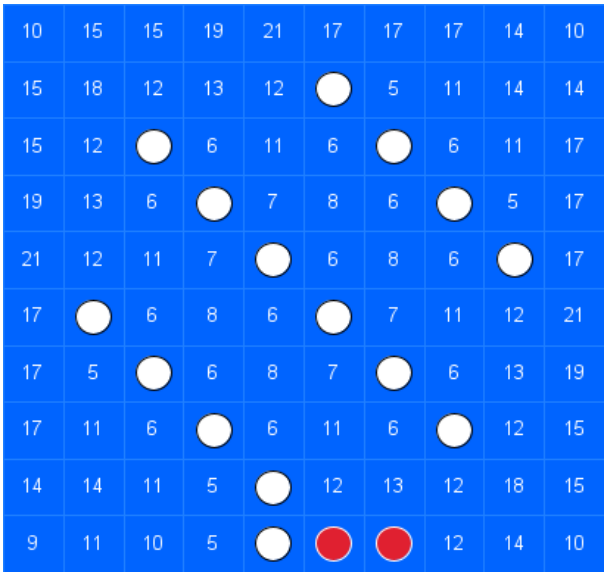
Αρχικά, όπως και προηγουμένως, ο αλγόριθμος ξεκινά με τη λειτουργία Hunt και ψάχνει το πιο πιθανό σημείο όπου μπορεί να βρίσκεται πλοίο. Μετά από αρκετά MISS, βρίσκει ένα σημείο πλοίου. Η λειτουργία Target ενεργοποιείται. Εδώ ξεκινά να εφαρμόζεται η νέα μας στρατηγική. Παίρνει τα 3 σημεία που είναι πιθανοί στόχοι (κυκλωμένα με πράσινο), και ψάχνει για κάθε σημείο, σε καθεμία από τις 4 κατευθύνσεις πόσα HIT υπάρχουν και τα μετράει. Για το αριστερό πιθανό σημείο,

με τον μετρητή 17, το μόνο σημείο HIT που υπάρχει, βρίσκεται δεξιά του. Οπότε, το σκορ για το κουτί 17 είναι 1 για δεξιά. Ομοίως, τα κουτιά 12 και 19 έχουν και αυτά 1 σκορ, κάτω και αριστερά αντίστοιχα. Έτσι, η στρατηγική θα επιστρέψει 3 σημεία, αφού και τα 3 έχουν ως μέγιστο σκορ το ίδιο, και ο αλγόριθμος στη συνέχεια θα επιλέξει ένα στην τύχη, και θα ρίξει βολή.

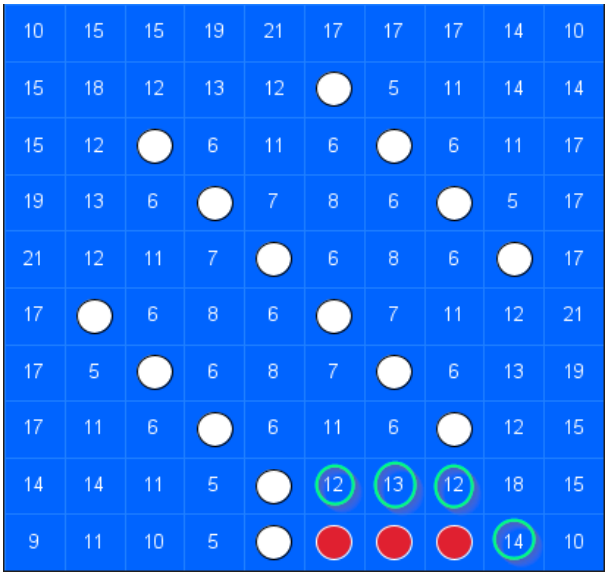


Επιλέγει λοιπόν το κουτάκι με μετρητή 19 και είναι και αυτό HIT. Ο αλγόριθμος προσθέτει τα νέα σημεία πιθανών στόχων στη λίστα (κουτιά 13 & 15), και τώρα εφαρμόζει ξανά την στρατηγική.

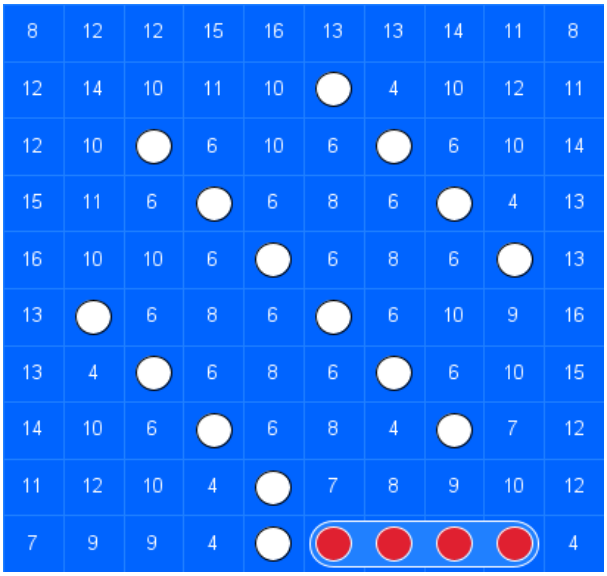
Αυτή τη φορά, τα σημεία 17 και 15 έχουν σκορ 2, ενώ τα σημεία 12 και 13 έχουν σκορ 1. Έτσι, τώρα επιστρέφει 2 σημεία, και ξανά ο αλγόριθμος θα επιλέξει τυχαία μεταξύ του κουτιού 17 και 15. Επιλέγει το κουτί 17 με MISS.



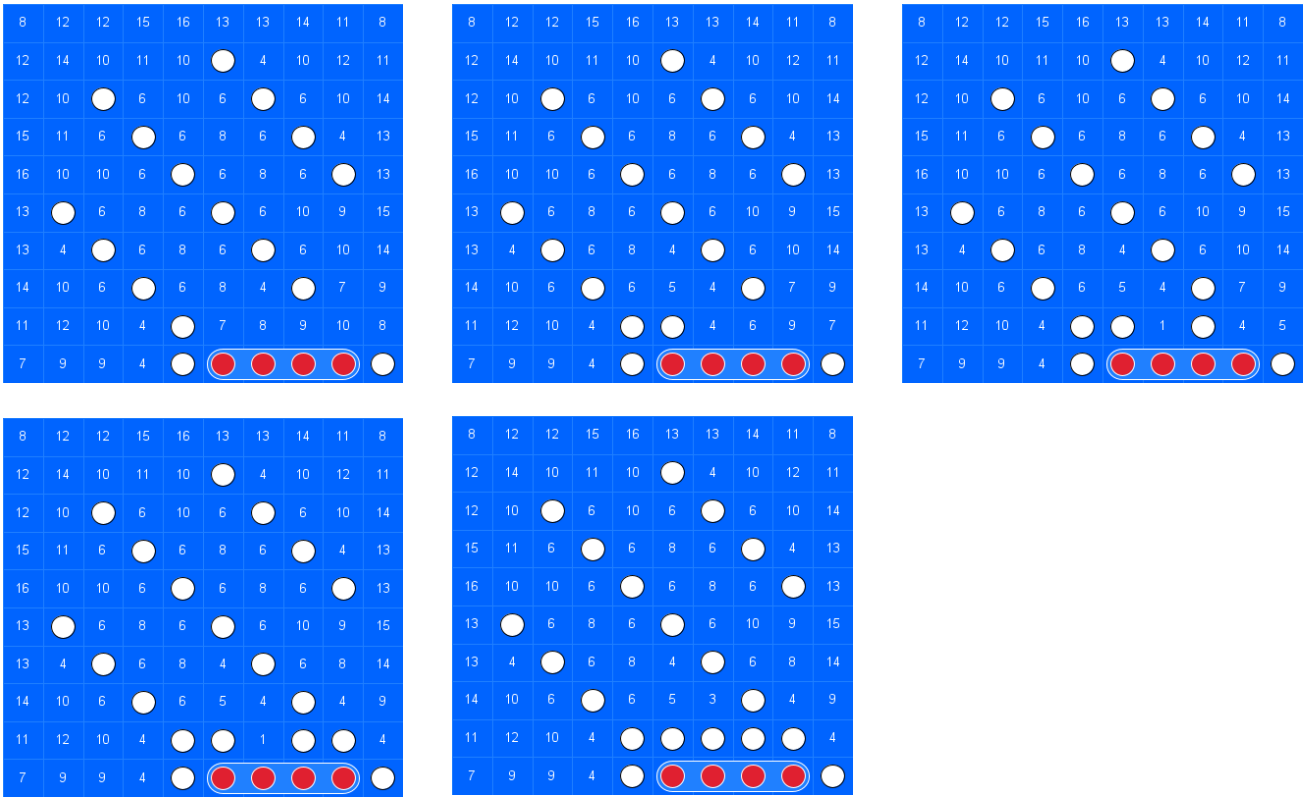
Εφαρμόζοντας την στρατηγική πάλι, αυτή τη φορά ο αλγόριθμος θα επιστρέψει ένα σημείο προς ρίψη βολής, καθώς μόνο το κουτί 15 έχει σκορ 2. Αυτή τη φορά έχουμε HIT.



Εδώ το κουτί 14 έχει το μέγιστο σκορ 3, και θα το επιλέξει ξανά για να ρίξει τη βολή.



Το πλοίο βρέθηκε ολόκληρο και βυθίστηκε, αλλά καθώς ο αλγόριθμος δε το ξέρει, θα συνεχίσει την στρατηγική μέχρι να μην υπάρχουν άλλα πιθανά σημεία στόχων. Το κουτάκι 4 έχει σκορ 4, οπότε θα χτυπήσει αυτό έπειτα.

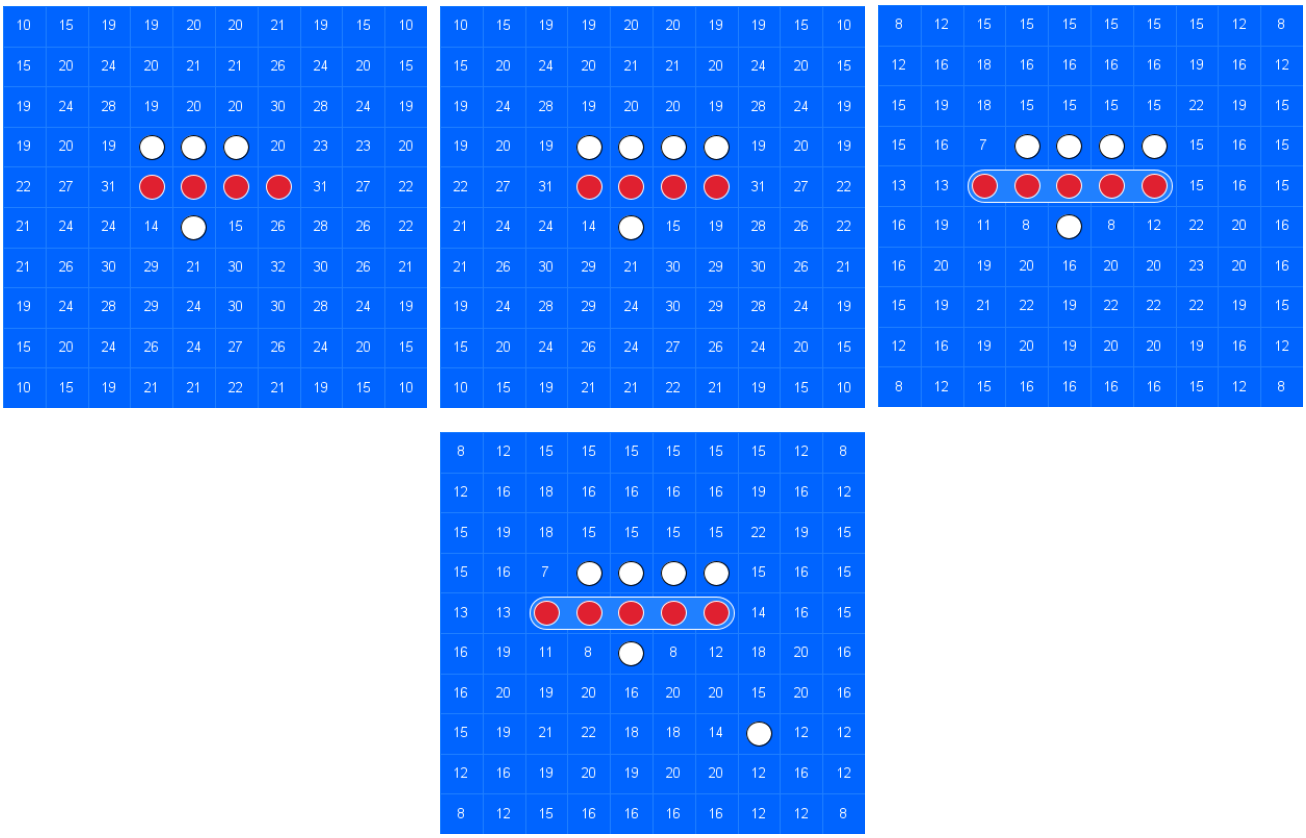


Στη συνέχεια, θα επισκεφτεί και όλα τα υπόλοιπα σημεία με σκορ 1.

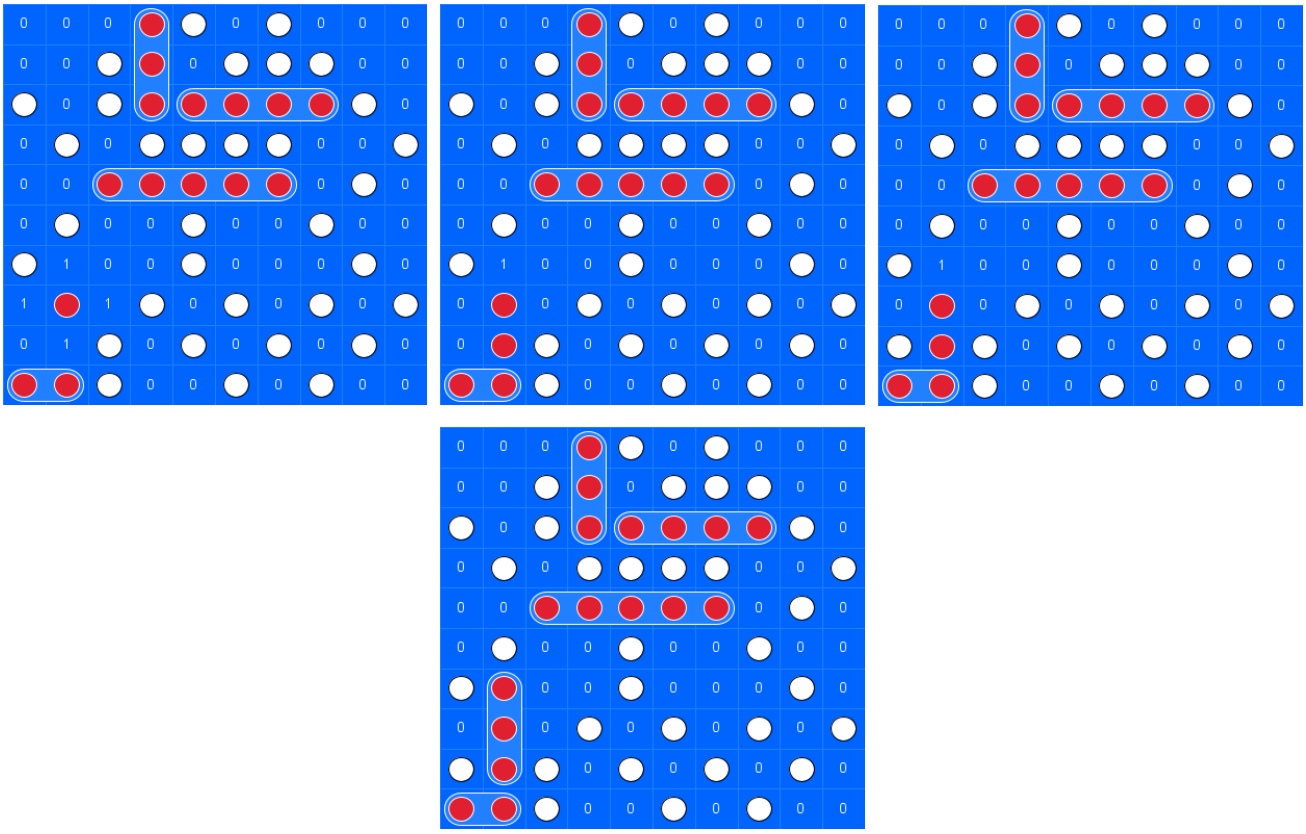
4.7 Στρατηγική “Ignore Sunken”

Για να μπορέσουμε πραγματικά να κατασκευάσουμε έναν αποδοτικό αλγόριθμο, πρέπει να μπορούμε να αναγνωρίσουμε καλύτερα πότε και αν ένα πλοίο έχει βυθιστεί. Οι προηγούμενες στρατηγικές του πράκτορα, αν και έξυπνες, αγνοούσαν εντελώς έναν κανόνα του παιχνιδιού, ο οποίος αναφέρει ότι είναι υποχρεωτική η ανακοίνωση βύθισης ενός πλοίου. Μόλις ο αλγόριθμος, λοιπόν, βυθίσει ένα πλοίο, το «αγνοεί» εντελώς, και δεν επισκέπτεται τα πιθανά σημεία της λίστας, όπως είχαμε αναφέρει προηγουμένως, γλυτώνοντας έτσι πολύτιμες κινήσεις.

Παρακάτω ακολουθούν μερικά παραδείγματα της στρατηγικής.



Βλέπουμε ότι μέχρι να βυθίσει όλο το πλοίο, ο αλγόριθμος συμπεριφέρεται ως γνωστόν. Μόλις το βυθίσει, ενώ προηγουμένως θα έπρεπε να επισκεφτεί όλα τα γειτονικά σημεία του πλοίου στη λίστα, τώρα τα αγνοεί και μπαίνει σε λειτουργία Hunt, ψάχνοντας το επόμενο πλοίο. Στο συγκεκριμένο παράδειγμα, ο αλγόριθμος γλύτωσε 7 επιπλέον κινήσεις.



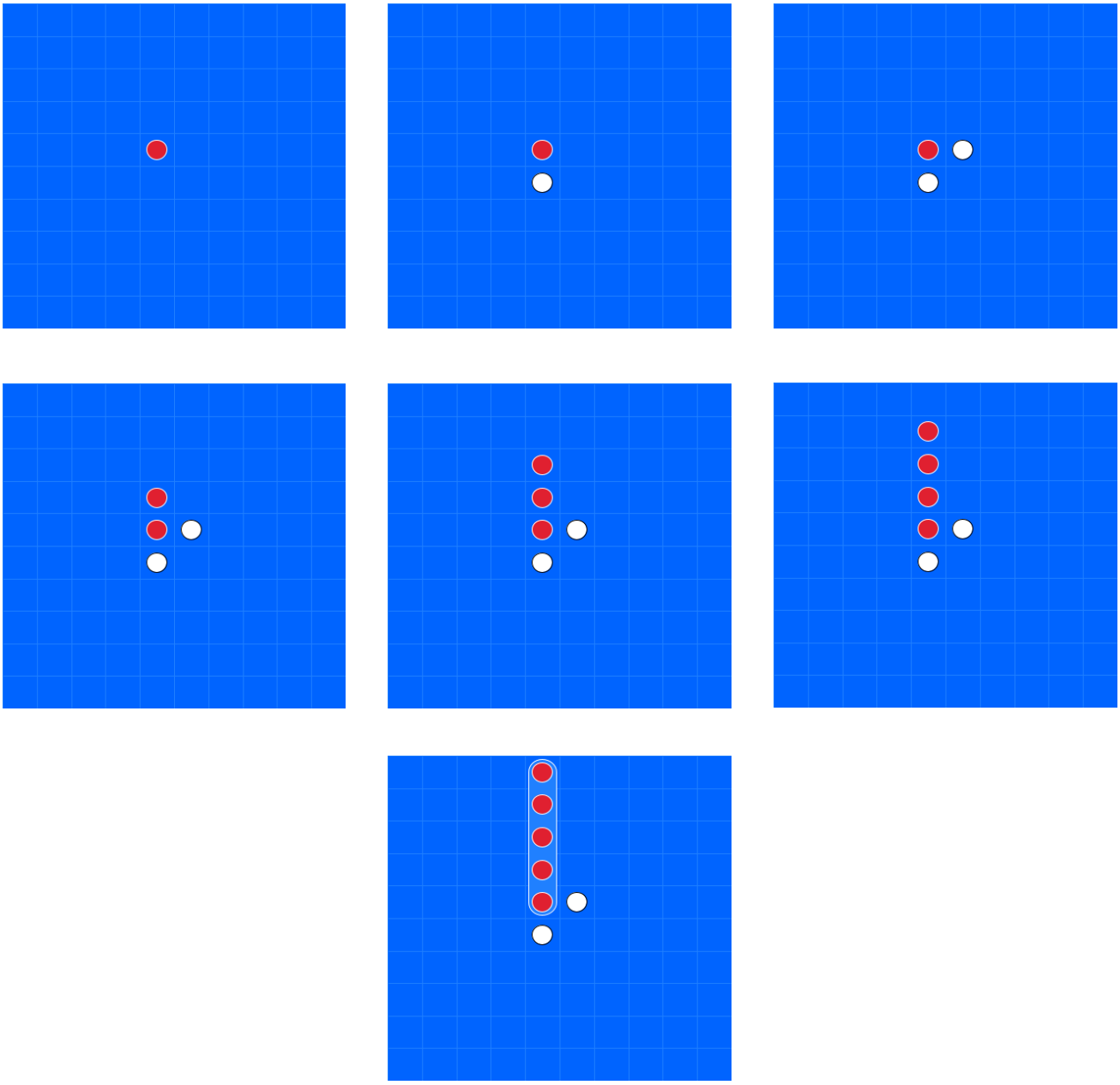
Τελικά, ολοκληρώνει το παιχνίδι σε 51 γύρους.

Ακολουθεί άλλο ένα παράδειγμα της στρατηγικής που μας βεβαιώνει για την χρησιμότητά της. Στο ίδιο παιχνίδι, για το πλοίο με μέγεθος 3, ο αλγόριθμος γλυτώνει 3 επιπλέον κινήσεις, οι οποίες θα καθυστερούσαν να λήξει και το παιχνίδι, καθώς είναι το τελευταίο πλοίο. Συνολικά, στο παιχνίδι αυτό, η στρατηγική “Ignore Sunken” μας γλύτωσε 13 ολόκληρες κινήσεις.

4.8 Συνδυασμός στρατηγικών “Target Line” & “Ignore Sunken”

Εφόσον έχουμε δει τις δύο αυτές στρατηγικές μόνες τους, μία καλή ιδέα είναι να τις συνδυάσουμε, για ακόμα καλύτερα αποτελέσματα. Όπως είδαμε στο πρώτο παράδειγμα της στρατηγικής “Ignore Sunken”, αν και γλυτώσαμε 7 κινήσεις από την βύθιση του aircraft carrier, θα μπορούσαμε πιθανότατα να γλυτώσουμε μερικές ακόμη, αν ο αλγόριθμος έριχνε βολές στην ευθεία, αντί στα πιθανά σημεία της λίστας.

Η λειτουργία του συνδυασμού δεν έχει τίποτα καινούριο, παρά μόνο την ταυτόχρονη δράση των δύο στρατηγικών.



Ο αλγόριθμος δεν ξοδεύει κινήσεις ψάχνοντας γύρω από το πλοίο για να σιγουρευτεί πλέον, και ρίχνει βολές στην ευθεία, τελειώνοντας έτσι πολύ γρήγορα την βύθιση του μεγαλύτερου πλοίου στο παιχνίδι.

Με παρόμοιο τρόπο βρίσκει και τα υπόλοιπα πλοία, για να λήξει το παιχνίδι στις 47 κινήσεις.

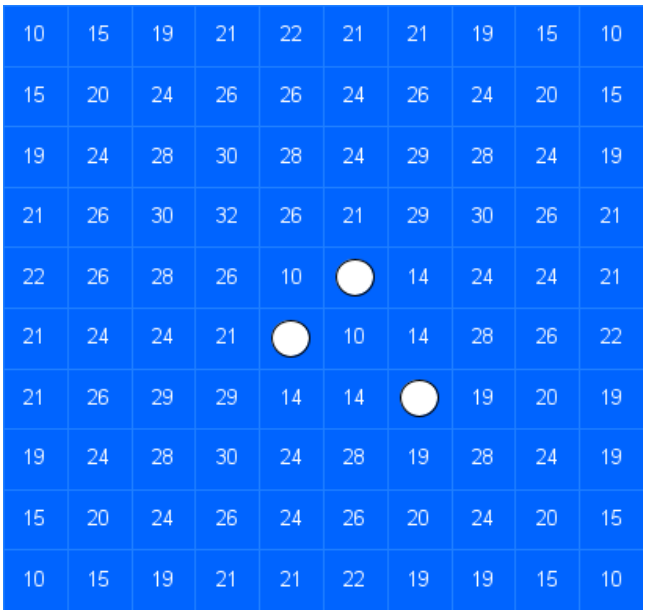
4.9 Στρατηγική Probability with Bomb Bonus

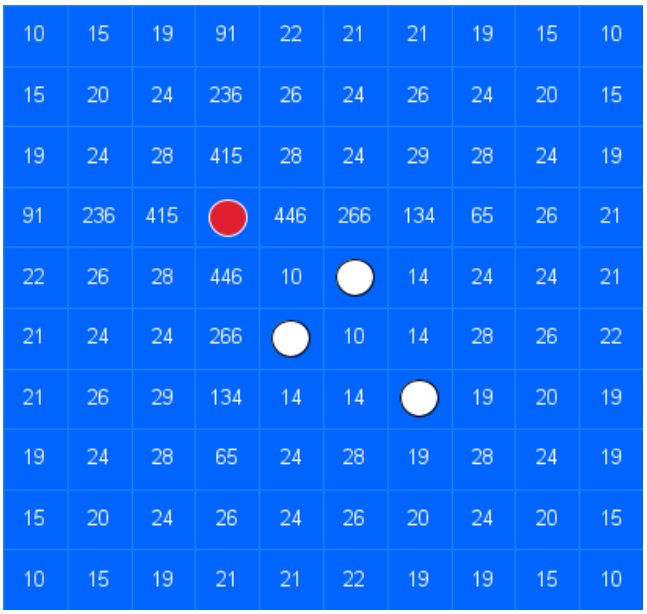
Ο πράκτορας μας έχει φτάσει σε μία πολύ καλή κατάσταση, όπου με τους κανόνες και τις στρατηγικές μας τον έχουμε κάνει πολύ αποδοτικό. Όμως, μπορούμε συνολικά να βελτιώσουμε ακόμα λίγο τον πράκτορά μας με την στρατηγική "Probability with Bomb Bonus". Ο αλγόριθμος έχει ακόμα λειτουργία Hunt και Target, που όμως και οι δύο λειτουργούν ουσιαστικά με τον ίδιο τρόπο.

Στη λειτουργία Hunt, ψάχνουμε με βάση τις πιθανότητες να βρούμε το πιο πιθανό σημείο που μπορεί να βρίσκεται ένα πλοίο και μας ενδιαφέρουν μόνο 3 πράγματα: Τα MISS, τα βυθισμένα πλοία, και τον χώρο που δεν έχουμε επισκεφτεί ακόμα. Τα MISS και τα βυθισμένα πλοία, όπως και προηγουμένως τα αντιμετωπίζουμε σαν εμπόδια, από τα οποία 100% δεν μπορεί να περνάει κάποιο νέο πλοίο από τα σημεία αυτά.

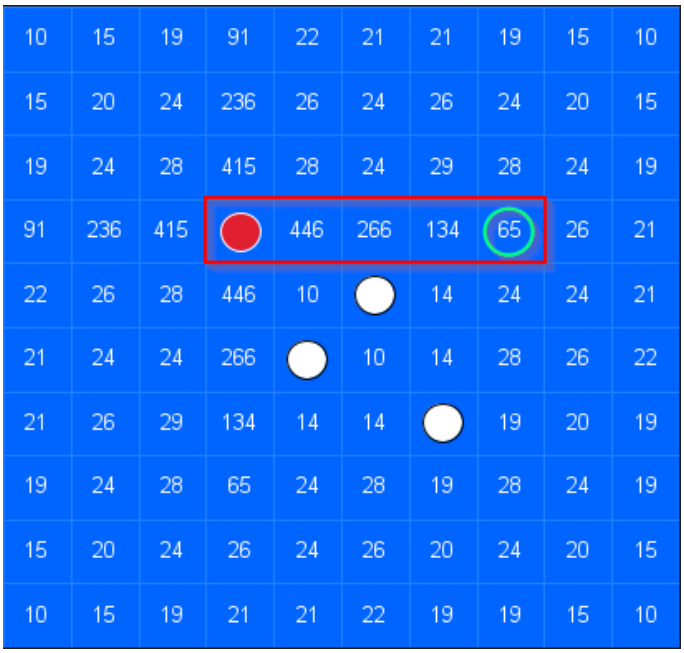
Στην λειτουργία Target, με δεδομένο τουλάχιστον ένα HIT, παίρνουμε όλα τα πλοία που δεν έχουν βυθιστεί ακόμη, και τρέχουμε επαναλήψεις στο πώς μπορούμε να τοποθετήσουμε τα πλοία ενώ συμπεριλαμβάνουν το HIT αυτό. Για κάθε κελί που ανήκει στις πιθανές τοποθετήσεις των πλοίων αυτών, δημιουργούμε ένα Bomb Bonus, το οποίο υπολογίζεται από το πλήθος των HIT κελιών και μία τιμή, που την ορίσαμε ως το διπλάσιο συνολικό μήκος των αβύθιστων πλοίων και αυξημένο κατά 1: $Bomb\ Bonus\ = Συνολικό\ μήκος\ αβύθιστων\ πλοίων * 2 + 1$. Ο μετρητής του κελιού θα υπολογιστεί πολλαπλασιάζοντας το Bomb Bonus με το πλήθος των HIT που περιλαμβάνει η τοποθέτηση του πιθανού πλοίου, προσθέτοντας κάθε φορά και τον εαυτό του: $Square\ Possibilities = Square\ Possibilities + Hit\ Count * Bomb\ Bonus$.

Παρακάτω παρουσιάζουμε πώς ακριβώς υπολογίζονται οι νέοι μετρητές.

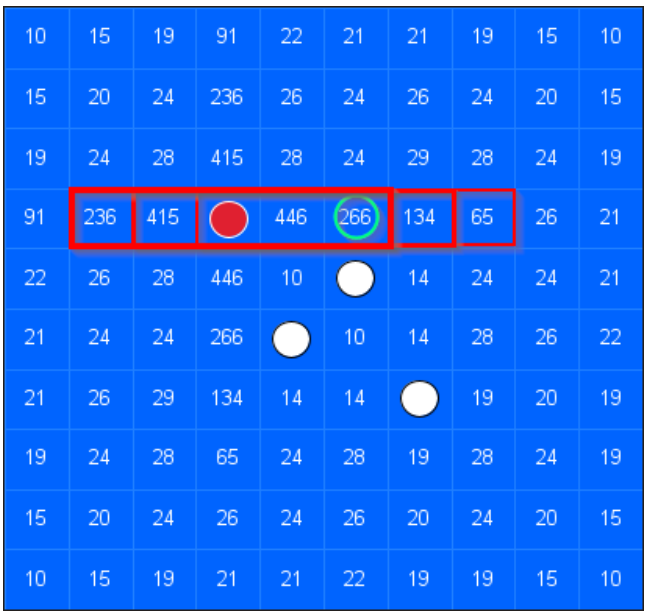




Βλέπουμε ότι μόλις ο αλγόριθμος βρει ένα HIT, αλλάζει τους μετρητές των κελιών όπου μπορούν να τοποθετηθούν πλοία, συμπεριλαμβάνοντας και το κελί με το HIT. Φυσικά, δεν έχουν όλα τα κελιά ίδιο αριθμό μετρητή, καθώς δεν μπορούν να περάσουν όλα τα πλοία από όλα κελιά. Για παράδειγμα, το κελί με μετρητή 65 περιέχεται μόνο στην πιθανή τοποθέτηση ενός Aircraft Carrier. Κανένα άλλο πλοίο δεν μπορεί να τοποθετηθεί εκεί, και να συμπεριλαμβάνει ταυτόχρονα και το κελί με μετρητή 65 και το HIT. Έτσι, με αρχικό μετρητή 30, το συγκεκριμένο κελί λαμβάνει Bonus 17*2 +1 = 35 και έτσι το τελικό άθροισμα είναι 30+35 = 65.

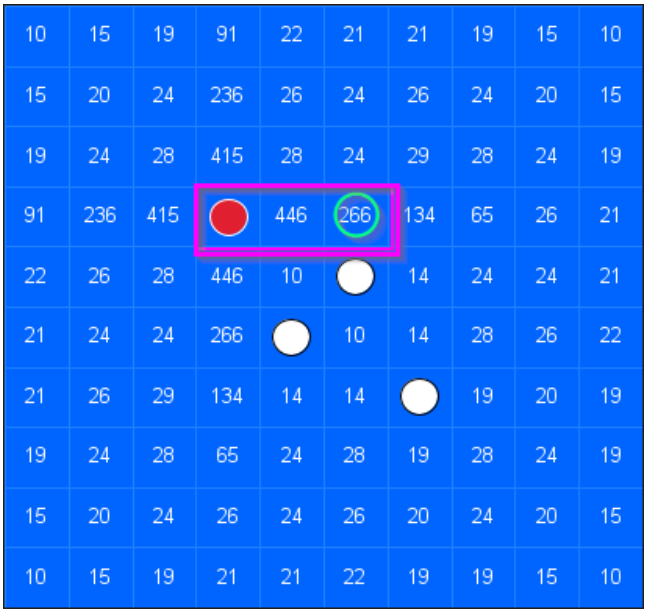


Το κελί με μετρητή 266 έχει περισσότερα πλοία που μπορούν να τοποθετηθούν περιλαμβάνοντας και το HIT. Ας τα δούμε.



Ξεκινάμε με το Aircraft Carrier. Βλέπουμε ότι υπάρχουν 3 πιθανές τοποθετήσεις του Aircraft Carrier που να έχουν ταυτόχρονα και το κελί 266 και το σημείο HIT. Ο μετρητής λοιπόν από 21 που ήταν αρχικά, αυξάνεται κατά $35 + 35 + 35 = 105$ ακόμα, σύνολο 126.

Για το Battleship, υπάρχουν μόνο 2 πιθανές θέσεις. Έτσι, έχουμε επιπλέον αύξηση κατά $35 + 35 = 70$, σύνολο 196 ως τώρα.



Τα πλοία Cruiser και Destroyer που έχουν μέγεθος 3, μπορούν από μία φορά το καθένα να τοποθετηθούν έτσι, ώστε να καλύπτουν το κελί και το HIT. Έτσι, τελικά, ο μετρητής μας έχει σύνολο $196 + 35 + 35 = 266$. Το Destroyer με μέγεθος 2 δεν μπορεί να περάσει από το HIT και από το κελί 266, οπότε το αγνοούμε στον υπολογισμό.

Με τον ίδιο ακριβώς τρόπο, υπολογίζονται όλα τα κελιά που βρίσκονται κοντά στο HIT.

Έτσι, ο αλγόριθμος βάζει περισσότερο βάρος στα σημεία όπου μπορούν να περάσουν τα περισσότερα πιθανά πλοία, αντί απλώς να χτυπάει στην ευθεία όπως έκανε η προηγούμενη στρατηγική. Όπως θα δούμε στα αποτελέσματα, η στρατηγική αυτή αποδίδει καλύτερα κατά μέσο όρο συγκριτικά με την προηγούμενη στρατηγική.

4.10 Στρατηγική Parity Probability with Bomb Bonus

Τέλος, δοκιμάσαμε να προσθέσουμε και τον περιορισμό Parity στην στρατηγική με το bomb bonus, που όμως βελτίωσε ελάχιστα τον μέσο όρο κινήσεων, όπως θα δούμε και στη συνέχεια.

4.11 Μία διαφορετική στρατηγική

Ως τώρα, όσες στρατηγικές αναλύσαμε, αφορούσαν το κομμάτι της εύρεσης των πλοίων μέσα στο board. Ουσιαστικά, όλες αυτές οι στρατηγικές βελτιώνουν την μία πλευρά του παιχνιδιού, που είναι η αναζήτηση και βύθιση των πλοίων της Ναυμαχίας. Όμως, μπορούμε να σκεφτούμε λίγο και την «απέναντι» πλευρά του παιχνιδιού, όπου στόχος είναι η καλή τοποθέτηση των πλοίων, ώστε να τα κρύψουμε όσο το δυνατόν καλύτερα.

Αναφορικά με αυτό, λοιπόν, δημιουργήσαμε μία στρατηγική “Anti-Target Placement”, η οποία όταν είναι ενεργοποιημένη, τοποθετεί το κάθε πλοίο με απόσταση τουλάχιστον μιας σειράς ή μιας στήλης, ώστε να μην εφάπτονται. Καθώς, οι περισσότερες στρατηγικές που έχουμε αναλύσει έχουν ως βάση την φιλοσοφία του “Hunt / Target” αλγόριθμου, τοποθετώντας τα πλοία μακριά το ένα με το άλλο, οι κινήσεις που χρειάζονται για να ολοκληρωθεί το παιχνίδι αυξάνονται, το οποίο επιβεβαιώνεται και από τα αποτελέσματα μας με τις προσομοιώσεις που θα δείξουμε στη συνέχεια.

5 Αποτελέσματα

Για να συγκρίνουμε όλες τις στρατηγικές, υλοποιήσαμε στον κώδικα ένα κομμάτι για να έχουμε την δυνατότητα να τρέχουμε πολλές προσομοιώσεις. Στην περίπτωση μας, τρέξαμε 2000 προσομοιώσεις για κάθε στρατηγική, με και χωρίς την στρατηγική "Anti-Target Placement", σε board 10x10. Στο τέλος θα παρουσιάσουμε και δοκιμές που έγιναν σε board 20x20, οι οποίες επιβεβαίωσαν ξανά την βελτίωση που είχε ο αλγόριθμος μας καθώς τον εξελίσσαμε περισσότερο.

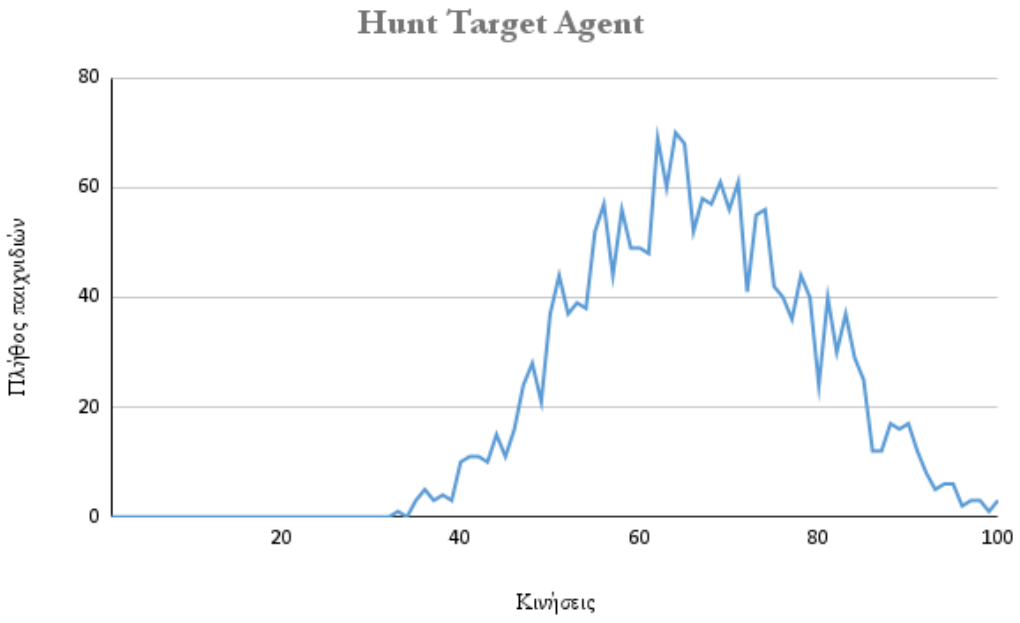
Για να γίνουν τα διαγράμματα κατανοητά και να γνωρίζουμε ποια καμπύλη αφορά κάποιον πράκτορα, σημειώνουμε με την παρακάτω λίστα τα εξής νούμερα:

- 1 = Random Agent
- 2 = Hunt/Target Agent
- 3 = Hunt/Target Parity Agent
- 4 = Hunt/Target Parity Probability Agent
- 5 = Hunt/Target Parity Probability with Target Line Agent
- 6 = Hunt/Target Parity Probability with Ignore Sunken Agent
- 7 = Hunt/Target Parity Probability with Target Line and Ignore Sunken Agent
- 8 = Probability with Bomb Bonus Agent
- 9 = Parity Probability with Bomb Bonus Agent

Παρουσιάζουμε τα διαγράμματα όλων των στρατηγικών σε board 10x10.

Σχήμα: 5-1

Όπως αναφέραμε και αρχικά, το να ρίχνουμε βολές τυχαία είναι η χειρότερη στρατηγική που θα μπορούσαμε να επιλέξουμε για να παίξουμε το παιχνίδι της Ναυμαχίας. Η πλειονότητα των παιχνιδιών χρειάστηκε και τις 100 κινήσεις για να λήξουν, ενώ ο ελάχιστος αριθμός κινήσεων που χρειάστηκε για να λήξει κάποιο παιχνίδι ήταν 66. Ο μέσος όρος κινήσεων της στρατηγικής είναι 95,419, το οποίο ήταν αρκετά αναμενόμενο, με τυπική απόκλιση 4,64.

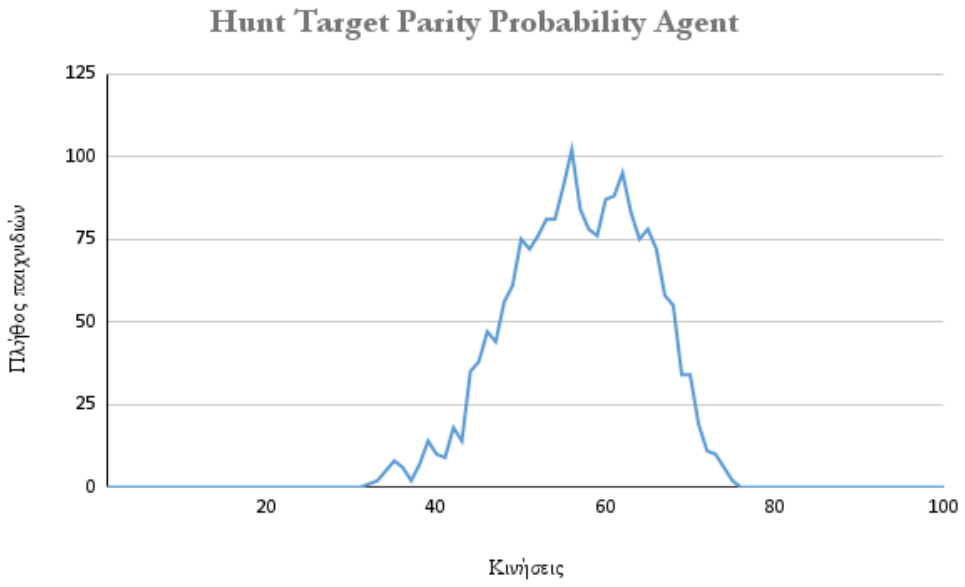


Σχήμα: 5-2

Εδώ έχουμε την πρώτη μεγάλη βελτίωση σε σχέση με τον τυχαίο πράκτορα, όπως φαίνεται και στο παραπάνω σχήμα. Παρόλο που υπήρχαν παιχνίδια που χρειάστηκαν 100 κινήσεις για να λήξουν, αυτά ήταν λίγα. Ο μέσος όρος κινήσεων είναι 66,09 με τυπική απόκλιση 12,46. Τέλος, το πιο γρήγορο παιχνίδι εδώ τελείωσε με 33 μόλις κινήσεις.

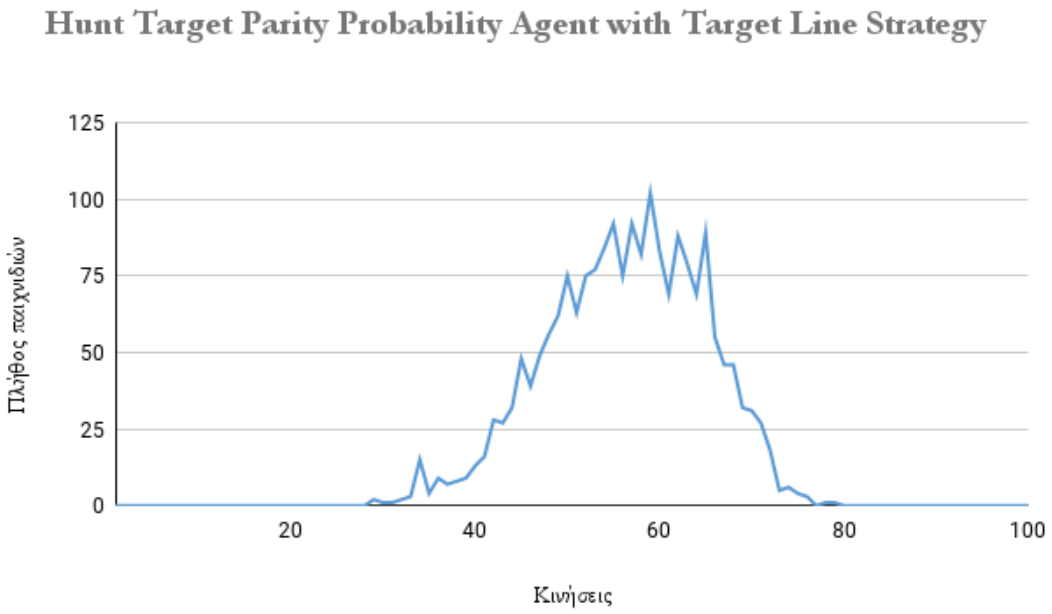
Σχήμα: 5-3

Η προσθήκη του περιορισμού Parity επιτέλους ρίχνει τον μέγιστο αριθμό κινήσεων που χρειάστηκε ένα παιχνίδι για να λήξει, από 100 σε 81 κινήσεις, και η βελτίωση φαίνεται επίσης και από τον μέσο όρο, ο οποίος έπεσε στις 61,27 κινήσεις, με τυπική απόκλιση 5,72. Μας φαίνεται λογικό, καθώς η χρησιμότητα να επιλέγουμε τις μισές θέσεις, αντί για το σύνολο, έριξε συνολικά τις κινήσεις που χρειάζεται ο αλγόριθμος για να λήξει ένα παιχνίδι.



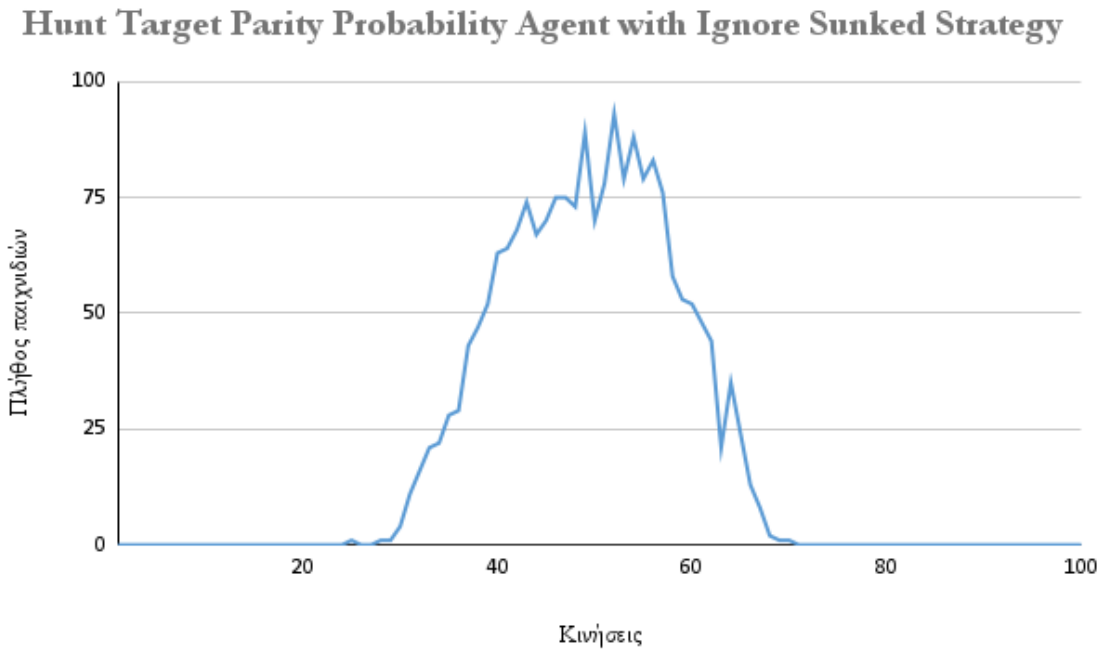
Σχήμα: 5-4

Μία από τις πιο βασικές προσθήκες, αν όχι η βασικότερη, είναι αυτή των πιθανοτήτων. Πλέον δεν ο αλγόριθμος δεν ρίχνει βολές τυχαία στη λειτουργία Hunt, αλλά υπολογίζει την πιο πιθανή θέση και ρίχνει εκεί. Βλέπουμε ότι η καμπύλη μετακινήθηκε ακόμα πιο αριστερά, με μέσο όρο τις 56,8 κινήσεις με τυπική απόκλιση 8. Τα παιχνίδια τελειώνουν πιο γρήγορα, και το χειρότερο παιχνίδι χρειάστηκε 75 βολές.



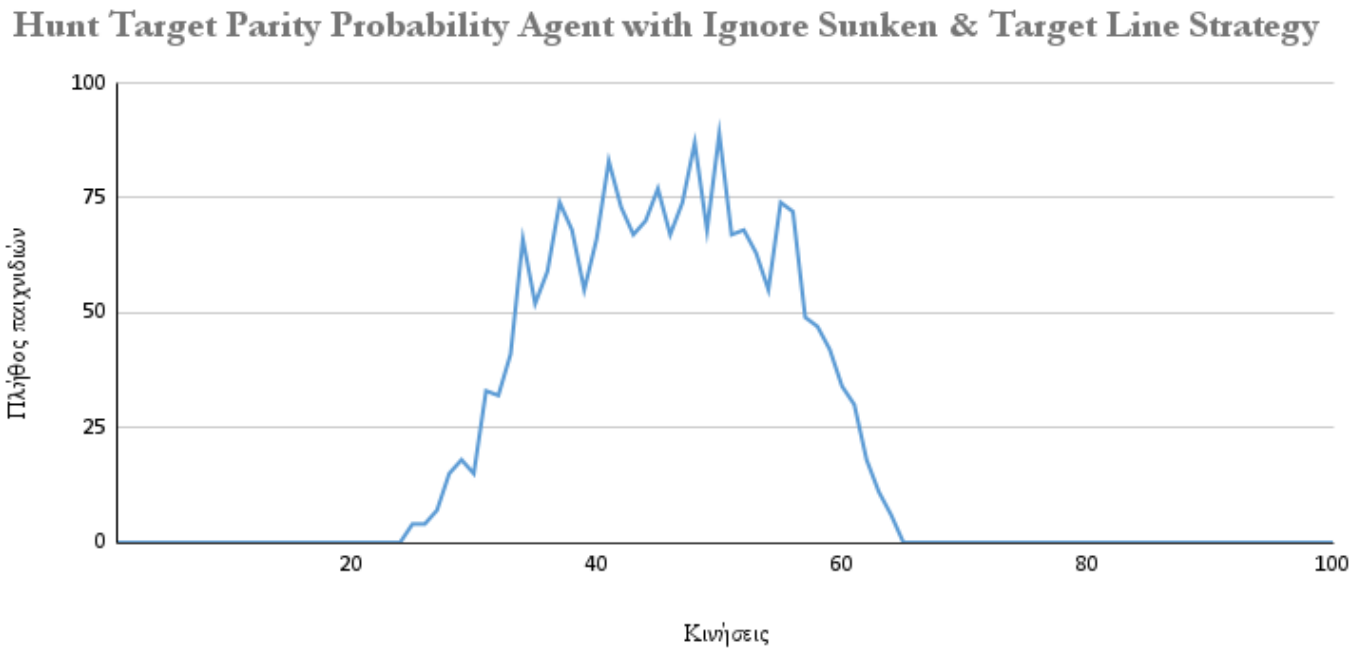
Σχήμα: 5-5

Η προσθήκη της στρατηγικής “Target Line” βελτίωσε ελάχιστα την απόδοση του αλγόριθμου μας. Καθώς χρειάζεται ακόμα να εξερευνήσει τα γειτονικά σημεία, η χρησιμότητά της δεν θα φανεί εδώ. Ο μέσος όρος είναι 56,2 κινήσεις με τυπική απόκλιση 8,49.



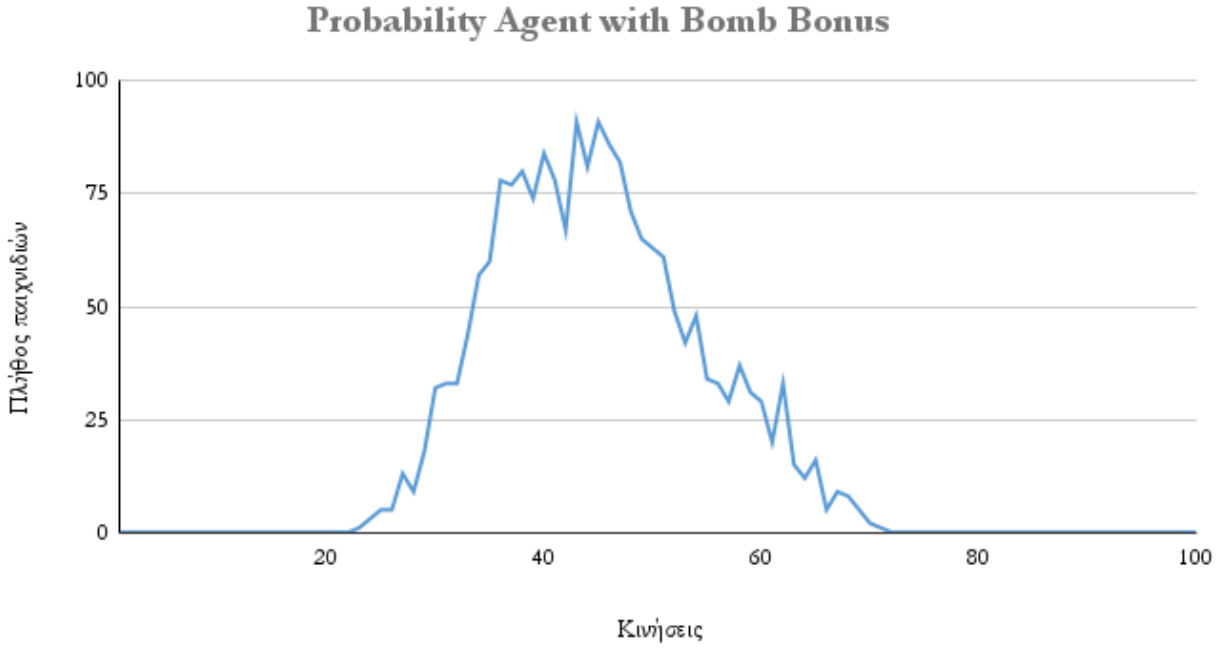
Σχήμα: 5-6

Εδώ έχουμε άλλη μία μεγάλη βελτίωση του αλγόριθμου, κάνοντας χρήση του κανόνα του παιχνιδιού που αναφέρει ότι είναι υποχρεωτική η ανακοίνωση βύθισης πλοίου. Έτσι, ο αλγόριθμος αγνοεί τα γειτονικά σημεία του βυθισμένου πλοίου, και αμέσως ψάχνει για νέο πλοίο. Με αυτή τη στρατηγική, μειώθηκαν επιπλέον οι κινήσεις για το χειρότερο και καλύτερο παιχνίδι. Το καλύτερο παιχνίδι της στρατηγικής έληξε με μόλις 25 κινήσεις, ενώ το χειρότερο με 70. Ο μέσος όρος κυμαίνεται στις 49,3 κινήσεις, με τυπική απόκλιση 8,39.



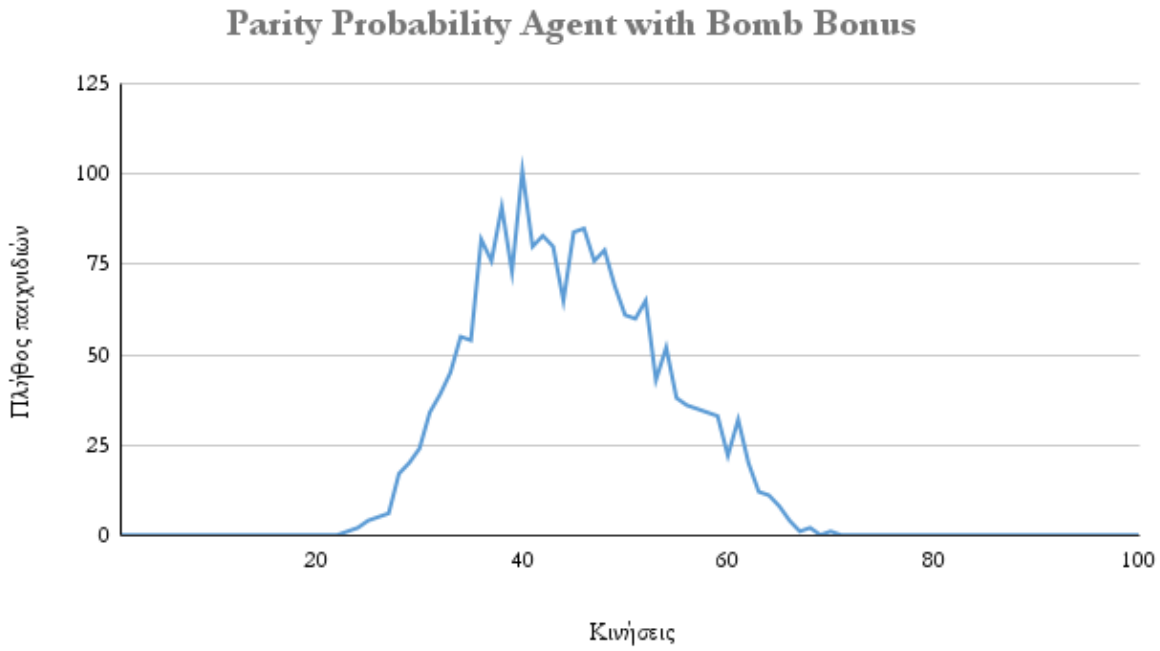
Σχήμα: 5-8

Η χρησιμότητα της στρατηγικής “Target Line” φαίνεται ξεκάθαρα στην περίπτωση αυτή, καθώς με τον συνδυασμό της στρατηγικής “Ignore Sunken”, ο αλγόριθμος δε θα ασχοληθεί με τα γειτονικά σημεία και παράλληλα θα βυθίσει το πλοίο πιο γρήγορα, ρίχνοντας βολές στην ευθεία. Ο μέσος όρος κινήσεων είναι στις 45,65 με τυπική απόκλιση 8,68. Παρατηρούμε ότι μειώθηκαν και οι κινήσεις που χρειάστηκε το χειρότερο παιχνίδι, από 70 που ήταν προηγουμένως στις 64.



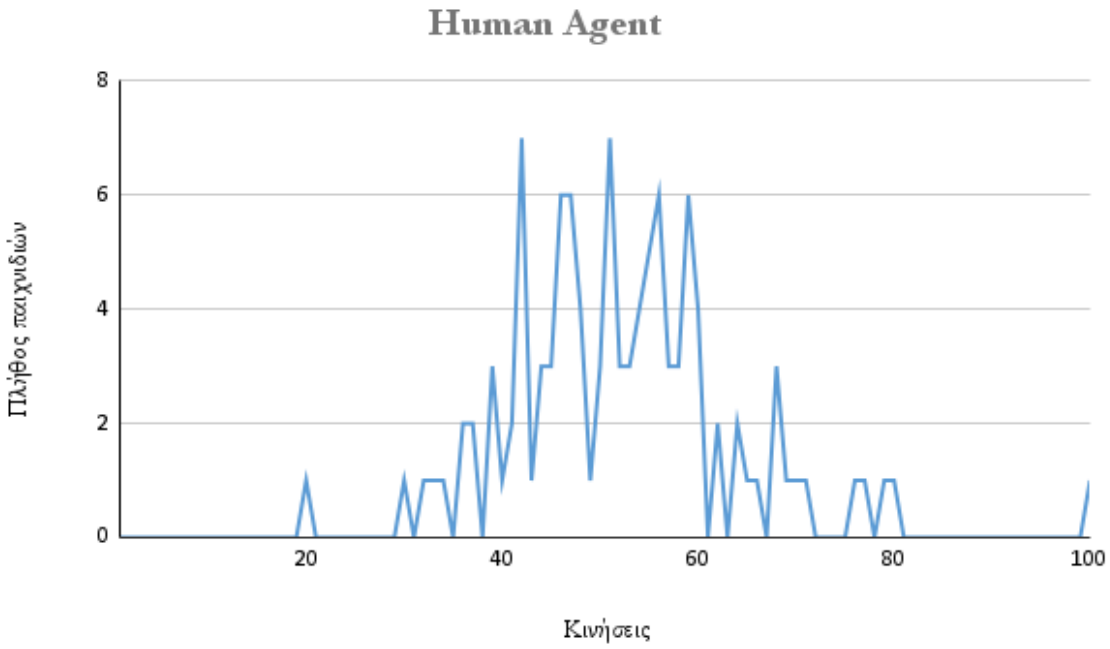
Σχήμα: 5-7

Ο αλγόριθμος μας εδώ έχει πάρει πλέον την τελική του μορφή. Αν και υπάρχει ακόμα λειτουργία Hunt & Target, πλέον αφομοιώθηκαν και λειτουργούν με τον ίδιο τρόπο, βάσει πιθανοτήτων και ενός bonus. Υπάρχει μία βελτίωση της τάξης της 1 κίνησης κατά μέσο όρο, ήτοι 44,84 κινήσεις με τυπική απόκλιση 9,18. Όπως παρατηρούμε από την κλίση της καμπύλης, υπάρχουν λιγότερα παιχνίδια που χρειάζονται πολλές κινήσεις.



Σχήμα: 5-9

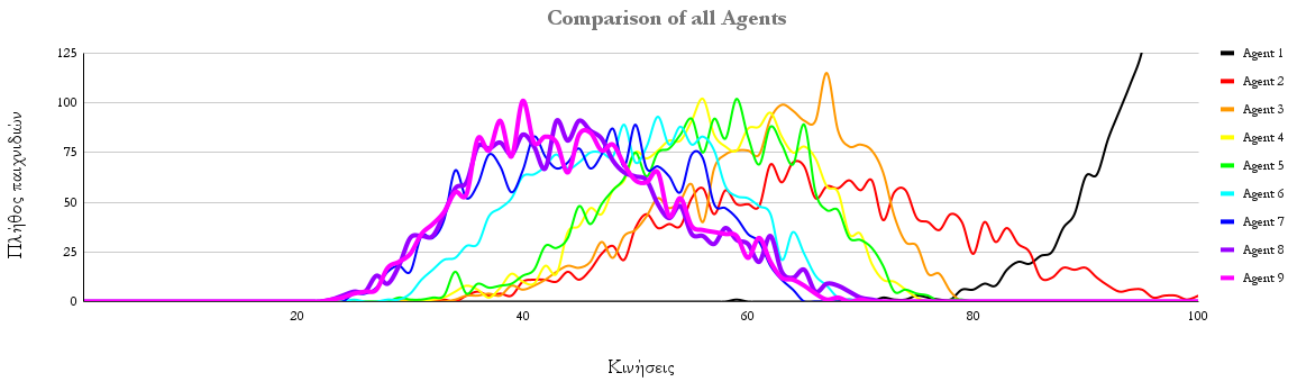
Ο περιορισμός Parity εδώ έχει ελάχιστη επίδραση, και βλέπουμε ότι η καμπύλη είναι σχεδόν ίδια. Ο μέσος όρος μειώθηκε κατά 0,3, στις 44,5 κινήσεις, και η τυπική απόκλιση είναι 8,73. Και στους δύο τελευταίους πράκτορες, βλέπουμε τα παιχνίδια με τις λιγότερες κινήσεις που χρειάστηκαν για να λήξουν, 23 και 21 αντίστοιχα. Αυτό είναι κάτι αξιοσημείωτο, ειδικά αν σκεφτεί κανείς ότι έχουμε υποχρεωτικές τουλάχιστον 17 κινήσεις, καθώς τόσο είναι το συνολικό άθροισμα των μεγεθών των πλοίων που έχουμε.



Σχήμα: 5-10

Τέλος, παρουσιάζουμε και τα συνδυαστικά αποτελέσματα ανθρώπων, ως έναν "ανθρώπινο πράκτορα", που έπαιξαν με τον κώδικα της εργασίας αυτής. Φυσικά, οι ανθρώπινες προσομοιώσεις δεν ήταν δυνατόν να αριθμούσαν το ίδιο πλήθος. Το παιχνίδι παίχτηκε 10-20 φορές από τον κάθε άνθρωπο που έλαβε μέρος στις δοκιμές, με συνολικό δείγμα 111 παιχνίδια από όλους τους εθελοντές. Από τα στοιχεία που μαζέψαμε, παρατηρήσαμε ότι ο μέσος όρος των κινήσεων ήταν στις 52,12 κινήσεις με τυπική απόκλιση 11,54. Αυτές οι μετρήσεις τοποθετούν τον ανθρώπινο παίκτη κάπου στη μέση της κατάταξης συγκριτικά με τους 9 πράκτορες της εργασίας, κάτι που το περιμέναμε.

Παρακάτω ακολουθεί και ένα συνολικό διάγραμμα με όλους τους πράκτορες μαζί.



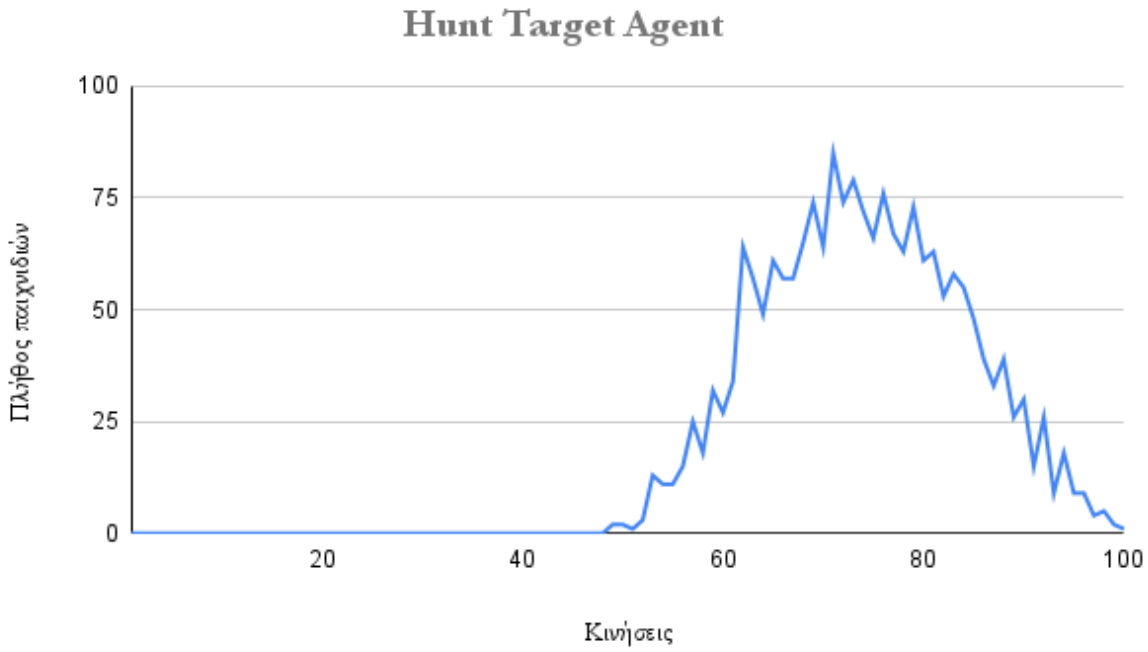
Σχήμα: 5-11

Παρατηρούμε πως με κάθε αλλαγή σχεδόν που πραγματοποιήσαμε στον πράκτορα μας, ο μέσος όρος κινήσεων, όπως και οι αντίστοιχες καμπύλες, μετακινούνται προς τα αριστερά, δηλαδή προς τις λιγότερες δυνατές κινήσεις. Είναι σαφές ότι οι 2 τελευταίοι πράκτορες με χρώματα μωβ και σκούρο ροζ αποδίδουν καλύτερα από όλους, αγγίζοντας σχεδόν τις 44 κινήσεις κατά μέσο όρο.

Στη συνέχεια, παρουσιάζονται τα διαγράμματα για το ταμπλό 10x10 με την στρατηγική “Anti-Target Placement” ενεργή.

Σχήμα: 5-12

Αν και η στρατηγική αυτή βάζει τα πλοία μακριά το ένα από το άλλο, δεν έχει καμία επίδραση στην χειρότερη στρατηγική που θα μπορούσαμε να σκεφτούμε για να παίξουμε το παιχνίδι της Ναυμαχίας. Το διάγραμμα έχει ακριβώς την ίδια μορφή, με μέσο όρο 95,42 κινήσεις και τυπική απόκλιση 4,84, νούμερα αρκετά παρόμοια με την περίπτωση απλής τοποθέτησης, χωρίς την επιπλέον στρατηγική.

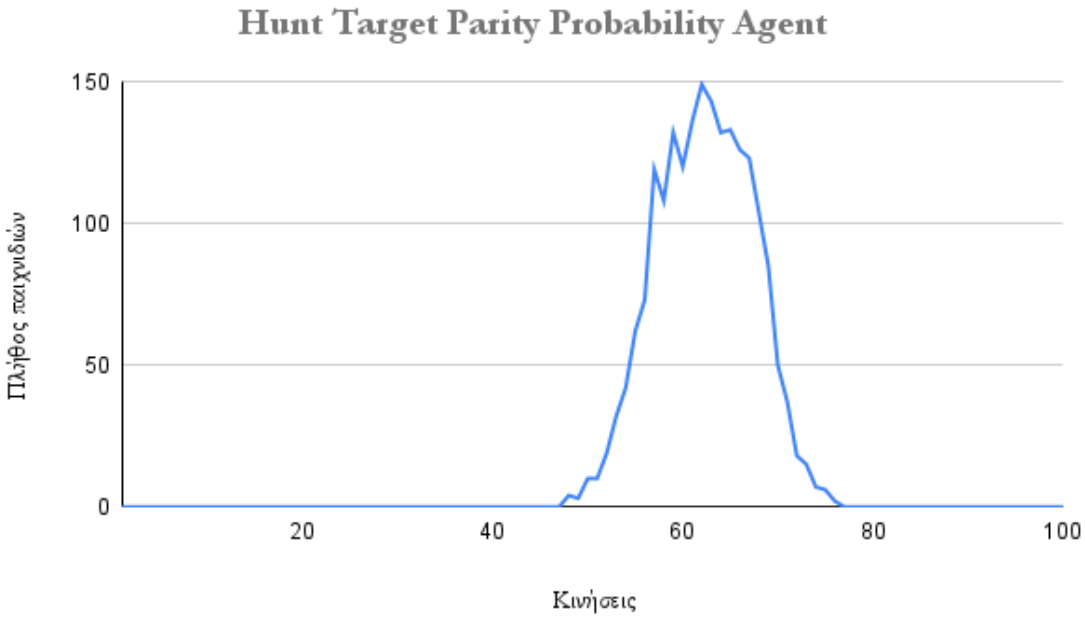


Σχήμα: 5-13

Το πρώτο θύμα της νέας στρατηγικής είναι ο Hunt Target πράκτορας. Η καμπύλη, αν και δεν άλλαξε μορφή, μετατοπίσθηκε προς τα δεξιά, αυξάνοντας τον μέσο όρο κινήσεων από 66,09 στις 74,07, με τυπική απόκλιση 9,78. Βλέπουμε λοιπόν ότι μία απλή στρατηγική που τοποθετεί τα πλοία μακριά το ένα από το άλλο, έκανε τον αλγόριθμο μας χειρότερο κατά 8 κινήσεις κατά μέσο όρο. Αυτό είναι και το μεγαλύτερο μειονέκτημα της υπάρχουσας λογικής, καθώς ψάχνει τυφλά γύρω-γύρω από όλα τα πλοία, για να σιγουρευτεί ότι βυθίστηκαν και ότι δεν υπάρχει άλλο πλοίο κοντά. Αν υπάρχει, τότε το βρίσκει γρήγορα και το βυθίζει. Αν όμως δεν υπάρχει, τότε, όπως δείξαμε και παραπάνω, καθυστερεί πάρα πολύ για να λήξει το παιχνίδι. Χαρακτηριστικό παράδειγμα είναι ότι το καλύτερο παιχνίδι του πράκτορα αυτού ήταν στις 49 κινήσεις, έναντι 33 κινήσεων χωρίς την “Anti-Target Placement”.

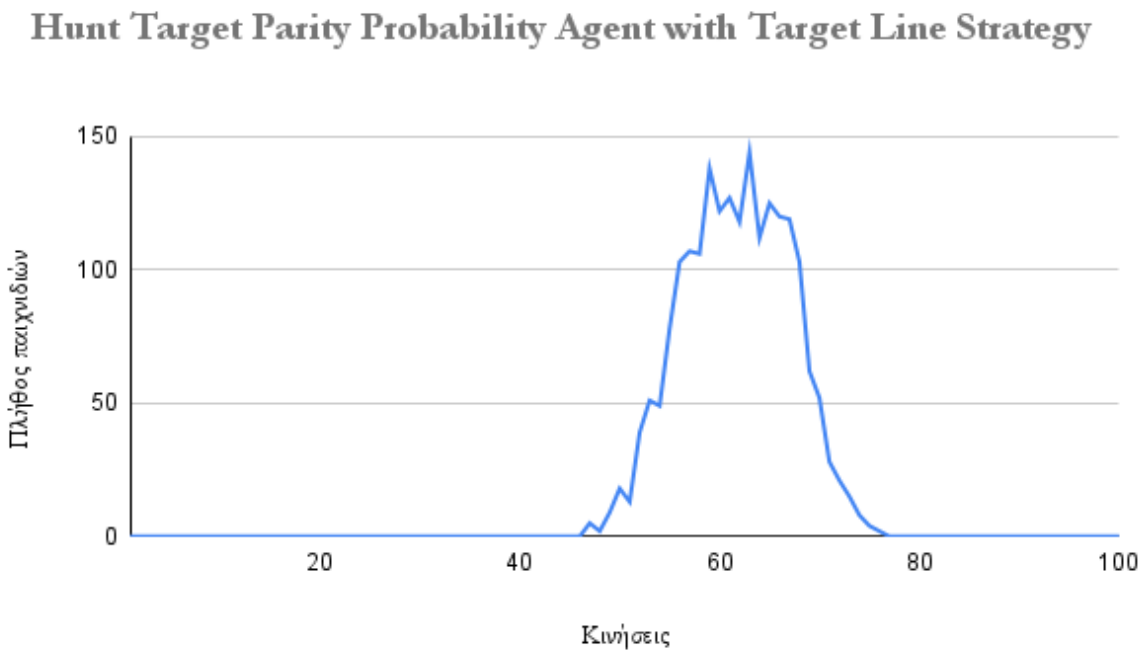
Σχήμα: 5-14

Η προσθήκη του Parity κάπως βελτιώνει τα πράγματα, αλλά είναι χειρότερο αποτέλεσμα συγκριτικά με τον πράκτορα χωρίς την "Anti-Target Placement" στρατηγική. Ο μέσος όρος εδώ είναι 67,9 κινήσεις (έναντι 61,2) με τυπική απόκλιση 5,72. Σαφώς και θα ήταν μια βελτίωση ο περιορισμός του Parity, καθώς παίρνει τα μισά κουτάκια, και όχι όλα προς έλεγχο. Όμως, και αυτή η στρατηγική αργεί πολύ, με ελάχιστο αριθμό κινήσεων για να λήξει ένα παιχνίδι στις 46 βολές, 3 μόλις λιγότερες από πριν.



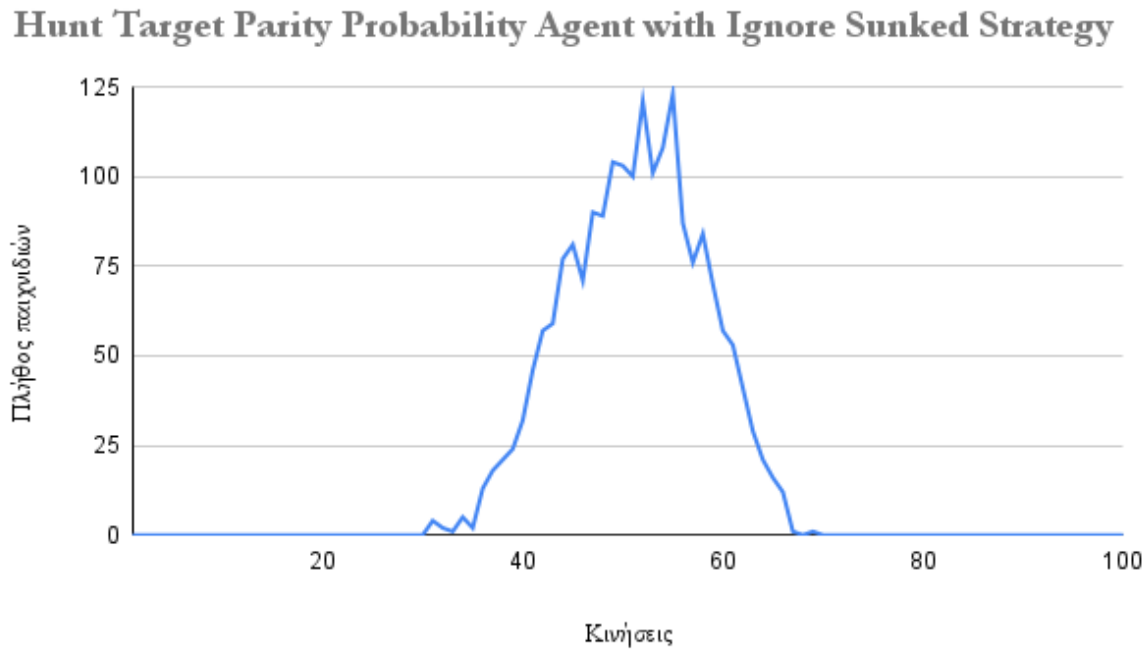
Σχήμα: 5-15

Η εισαγωγή των πιθανοτήτων, όπως ήταν αναμενόμενο, βελτίωσε κι άλλο τα αποτελέσματα, αλλά η νέα στρατηγική τοποθέτησης εξακολουθεί να κάνει χειρότερους τους αλγόριθμους μας. Ο μέσος όρος κινήσεων είναι 62,3 (έναντι 56,8) με τυπική απόκλιση 5. Αν και το χάσμα επιδόσεων μειώθηκε ήδη πριν με την προσθήκη του Parity, η στρατηγική με τις πιθανότητες το μειώνει λίγο ακόμα.



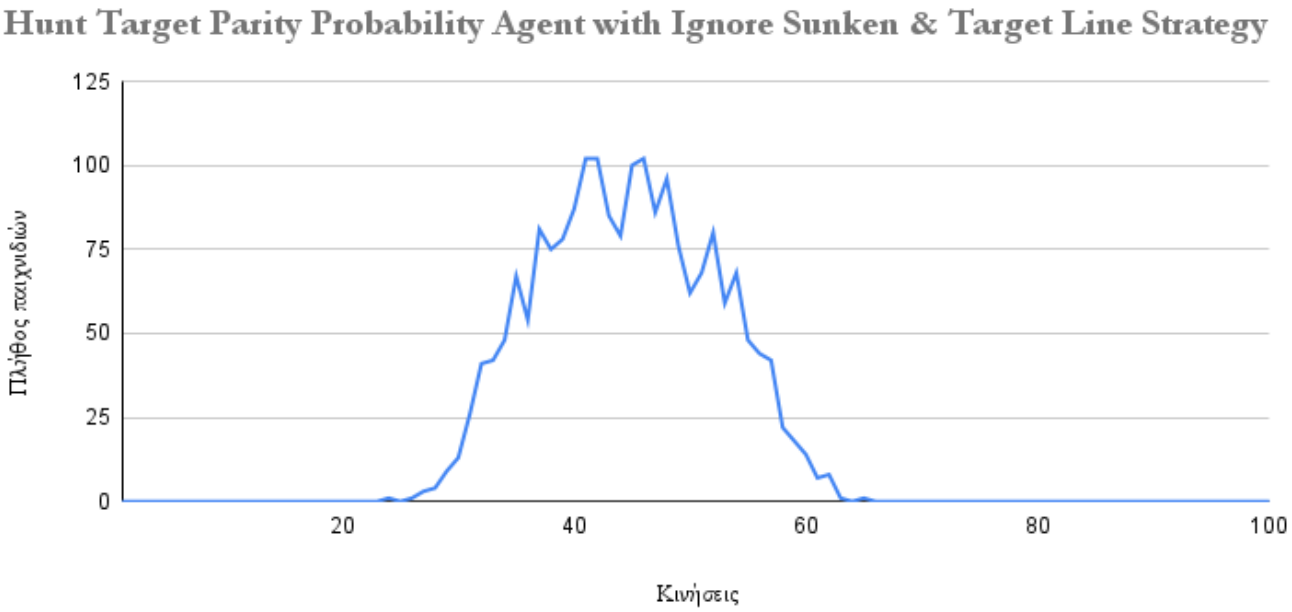
Σχήμα: 5-16

Όπως και με τον αντίστοιχο πράκτορα χωρίς την νέα στρατηγική, τα αποτελέσματα βελτιώθηκαν ελάχιστα με την στρατηγική “Target Line”. Η πλειονότητα των παιχνιδιών είναι συγκεντρωμένη γύρω από τον μέσο όρο, όπως φαίνεται από την καμπύλη, σε αντίθετη με την αντίστοιχη περίπτωση χωρίς την στρατηγική τοποθέτησης, όπου η καμπύλη είναι πιο «απλωμένη». Ο μέσος όρος είναι 61,68 με τυπική απόκλιση 5,3.



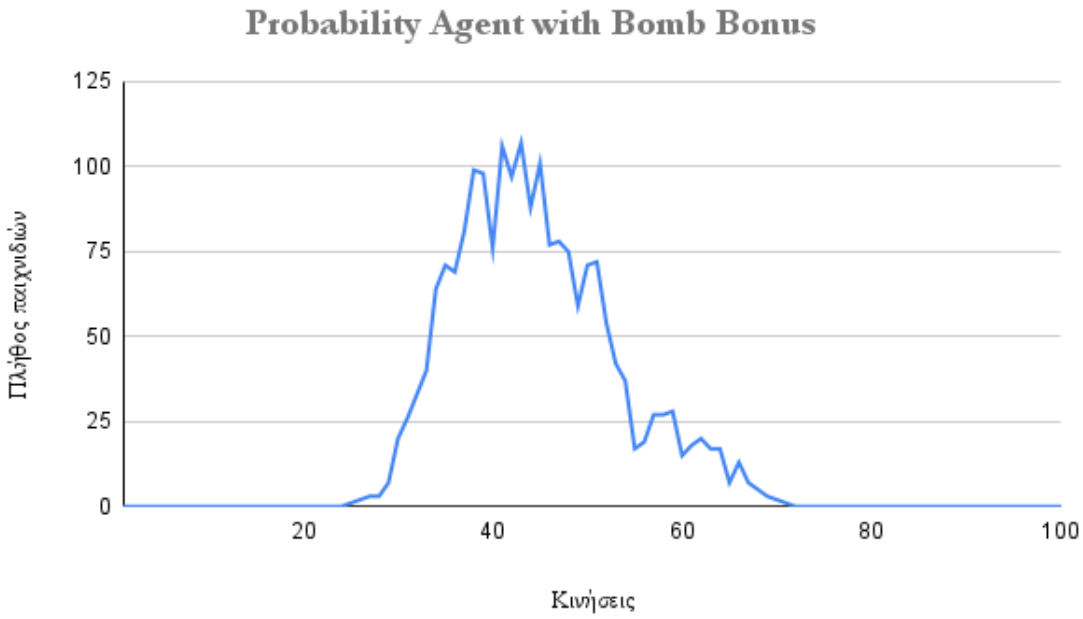
Σχήμα: 5-17

Η στρατηγική “Ignore Sunken” βελτιώνει αισθητά τα αποτελέσματα του αλγόριθμου μας, καθώς ρίχνει τον μέσο όρο κατά 10 κινήσεις, από 61,68 που ήταν προηγουμένως στις 51,17 κινήσεις, με απόκλιση 6,78. Αν και ακόμα έχουμε χειρότερα αποτελέσματα σε σχέση με τον αντίστοιχο πράκτορα χωρίς την τοποθέτηση, η διαφορά κινήσεων κατά μέσο όρο είναι σχεδόν 2. Καθώς η στρατηγική αυτή αγνοεί τα γειτονικά σημεία όταν βυθίσει ένα πλοίο, μας φαίνεται λογικό που η απόδοση της πλησιάζει αρκετά την απόδοση του αντίστοιχου πράκτορα χωρίς την τοποθέτηση.



Σχήμα: 5-18

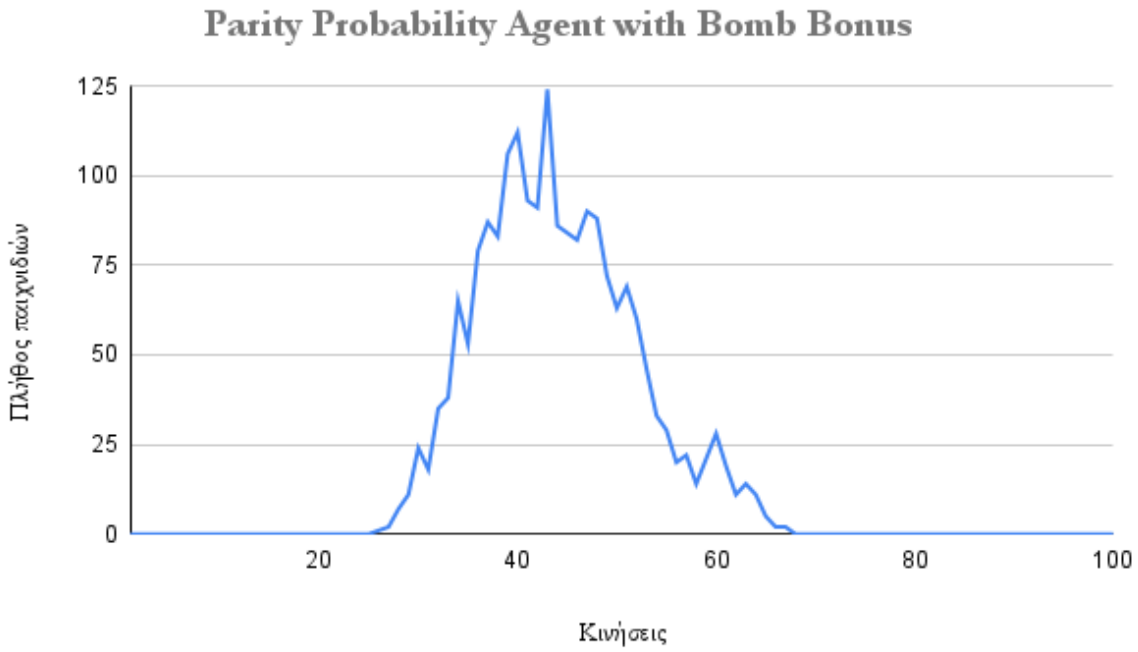
Ενδιαφέρον έχει ο επόμενος πράκτορας, που είναι ο συνδυασμός των δύο προηγούμενων στρατηγικών. Για πρώτη φορά, ο αλγόριθμος μας απέδωσε καλύτερα με την νέα στρατηγική τοποθέτησης συγκριτικά με τον αντίστοιχο αλγόριθμο χωρίς αυτήν. Ο μέσος όρος κινήσεων είναι στις 44,55 έναντι 45,65. Η διαφορά είναι μόλις μία κίνηση κατά μέσο όρο, αλλά είναι βελτίωση. Θα εξηγήσουμε στη συνέχεια τον λόγο.



Σχήμα: 5-19

Πλέον, ο αλγόριθμος μας αποδίδει σχεδόν το ίδιο καλά με και χωρίς την “Anti-Target Placement” στρατηγική, έχοντας μέσο όρο κινήσεων 44,53 με τυπική απόκλιση 8,38, τα οποία στατιστικά είναι αρκετά παρόμοια με τα στατιστικά του πράκτορα χωρίς την στρατηγική τοποθέτησης (44,84/9,18).

Μία μικρή βελτίωση βλέπουμε και στον τελευταίο μας πράκτορα, όταν προσθέσουμε το Parity.



Σχήμα: 5-20

Ο μέσος όρος κατέβηκε στις 44,07 κινήσεις με τυπική απόκλιση 7,74, όντας μισή κίνηση κατά μέσο όρο μικρότερος από τον αντίστοιχο πράκτορα χωρίς τη τοποθέτηση που είχε 44,51 κινήσεις κατά μέσο όρο.

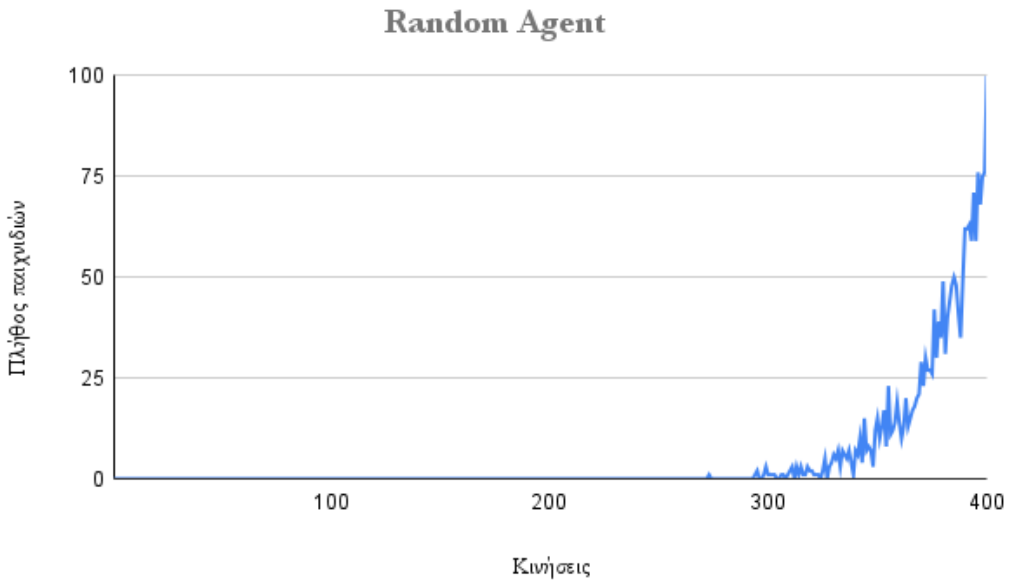
Ο λόγος που οι 3 τελευταίοι πράκτορες απέδωσαν καλύτερα τελικά με την “Anti-Target Placement” οφείλεται πιθανώς στο γεγονός ότι οι υπόλοιποι 6 πράκτορες καθυστερούσαν να λήξουν το παιχνίδι, και συγκεκριμένα καθυστερούσαν στο κομμάτι της λειτουργίας Target. Η λειτουργία Target έχει ως φιλοσοφία ότι ψάχνουμε πάνω, κάτω, δεξιά, αριστερά, από κάθε σημείο HIT. Έτσι, καθώς χρειάζονταν να κάνουν πολλές κινήσεις για να βεβαιωθούν ότι δεν υπάρχει κάποιο πλοίο κοντά, και με την νέα στρατηγική όντως δεν θα υπήρχε κανένα πλοίο κοντά, ο αλγόριθμος καθυστερούσε να λήξει το παιχνίδι, ξοδεύοντας περιττές κινήσεις. Αντιθέτως, οι πράκτορες χωρίς την τοποθέτηση έψαχναν πάλι τα γειτονικά σημεία για να βεβαιωθούν, αλλά κάποιες φορές έβρισκαν πλοία το ένα κοντά στο άλλο. Με αυτόν τον τρόπο, έληγαν το παιχνίδι πιο γρήγορα. Από τους 6 αυτούς πράκτορες εξαίρεση αποτελεί ο πράκτορας με την στρατηγική “Ignore Sunken”. Αυτός ο πράκτορας αγνοεί τα γειτονικά σημεία γύρω από ένα βυθισμένο πλοίο, και αυτός είναι και ο λόγος που η απόδοση του

πλησίασε τόσο πολύ την απόδοση του αντίστοιχου πράκτορα του. Όμως, δεν έβρισκε το πλοίο γρήγορα γιατί δεν έριχνε βολές στην ευθεία. Αυτός ακριβώς είναι και ο λόγος που ο 7ος πράκτορας με τις δύο συνδυασμένες στρατηγικές είχε μόλις 1 κίνηση κατά μέσο όρο λιγότερη από τον αντίστοιχο του. Δεν καθυστερούσε τόσο στη λειτουργία Target, τα βύθιζε γρήγορα, και έπειτα έψαχνε στη λειτουργία Hunt με βάση τις πιθανότητες. Εδώ πρέπει να σημειώσουμε όμως και έναν ακόμα λόγο που συντέλεσε στην καλύτερη απόδοσή του. Κάποια πλοία είναι πιθανό να βρίσκονται κάπου αλλού στο board αντί κοντά σε κάποιο ήδη βυθισμένο πλοίο. Έτσι, αν κάποια πλοία βρίσκονται το ένα δίπλα στο άλλο, και ένα βυθιστεί, ο αλγόριθμος θα ψάξει στο υπόλοιπο board και όχι στην γειτονιά του βυθισμένου, καθώς είναι πιο πιθανό να υπάρχουν περισσότερες πιθανές τοποθετήσεις πλοίων μακριά από το βυθισμένο, παρά κοντά του, αφού πλέον από τις θέσεις του βυθισμένου πλοίου δεν μπορεί να περνά κάποιο άλλο πλοίο. Έτσι, βάσει αυτών των δύο λόγων, ο 7ος πράκτορας είναι οριακά καλύτερος με την στρατηγική τοποθέτησης σε σχέση με τον ίδιο πράκτορα χωρίς αυτήν.

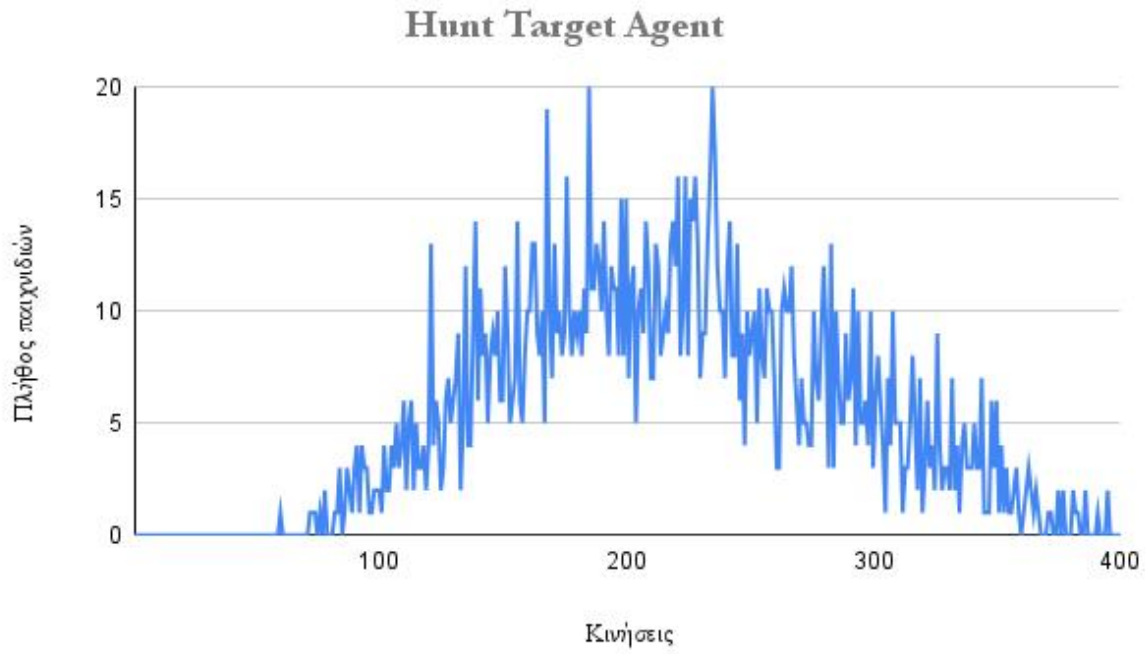
Τέλος, για τους ίδιους λόγους, οι πράκτορες 8 και 9 είναι πιο αποδοτικοί με την στρατηγική τοποθέτησης.

Στη συνέχεια, παρουσιάζουμε τα αντίστοιχα διαγράμματα για την περίπτωση ενός board 20x20.

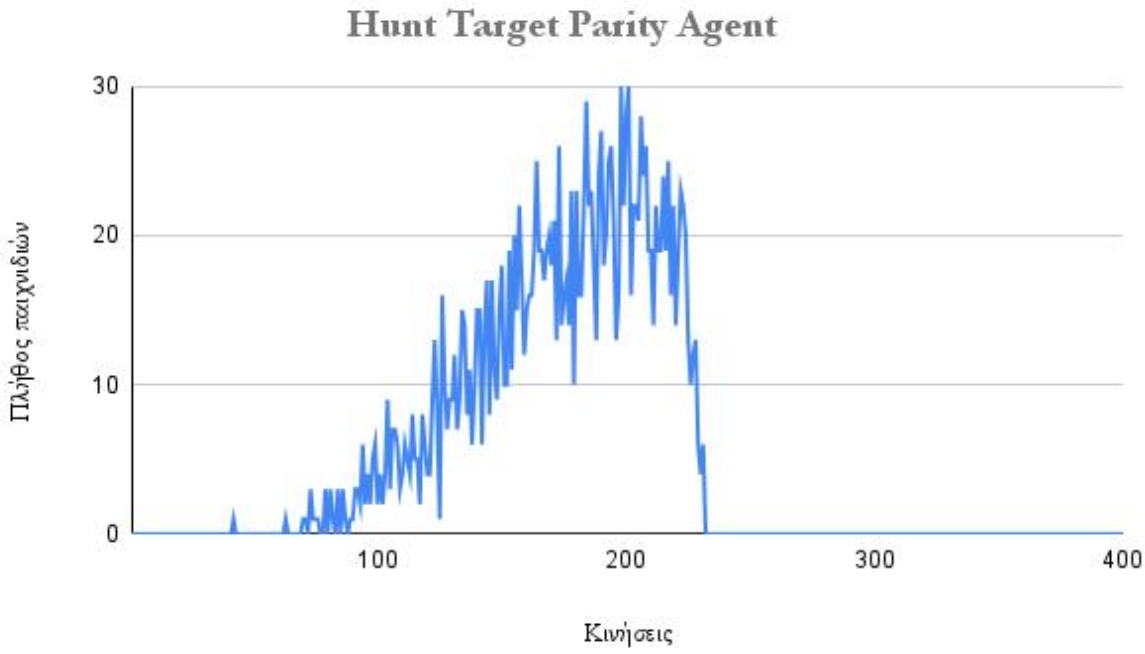
Εδώ το σύνολο των θέσεων είναι 400 και ο εντοπισμός των πλοίων είναι πολύ πιο δύσκολος, καθώς διατηρήσαμε τον ίδιο στόλο (5 πλοία που καταλαμβάνουν συνολικά 17 θέσεις).



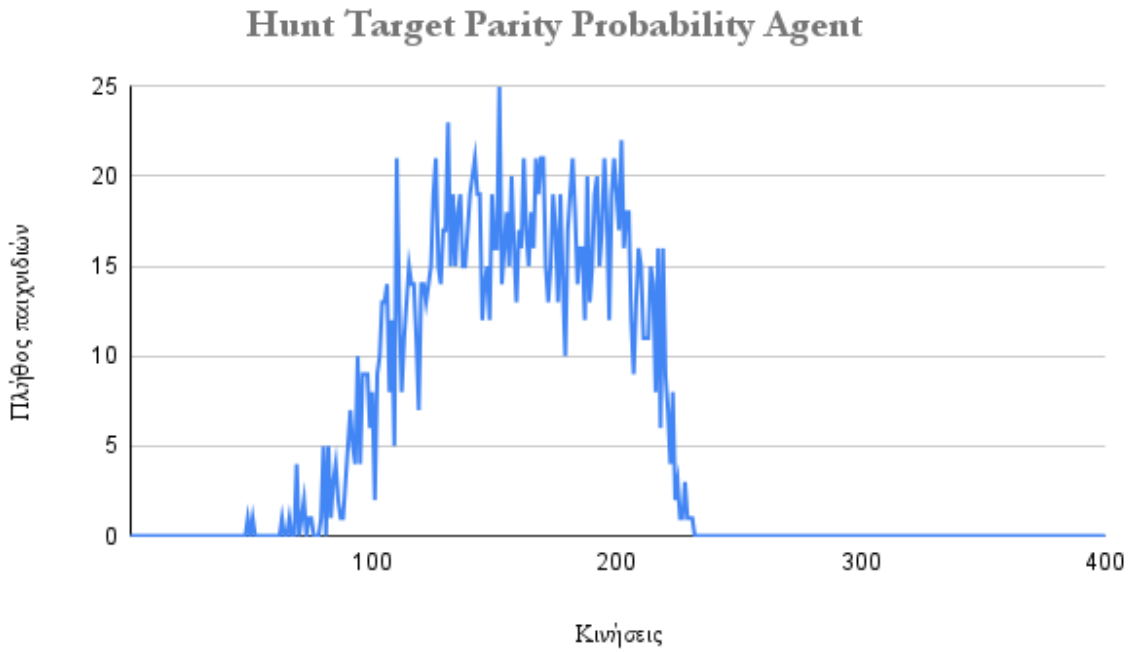
Σχήμα: 5-21



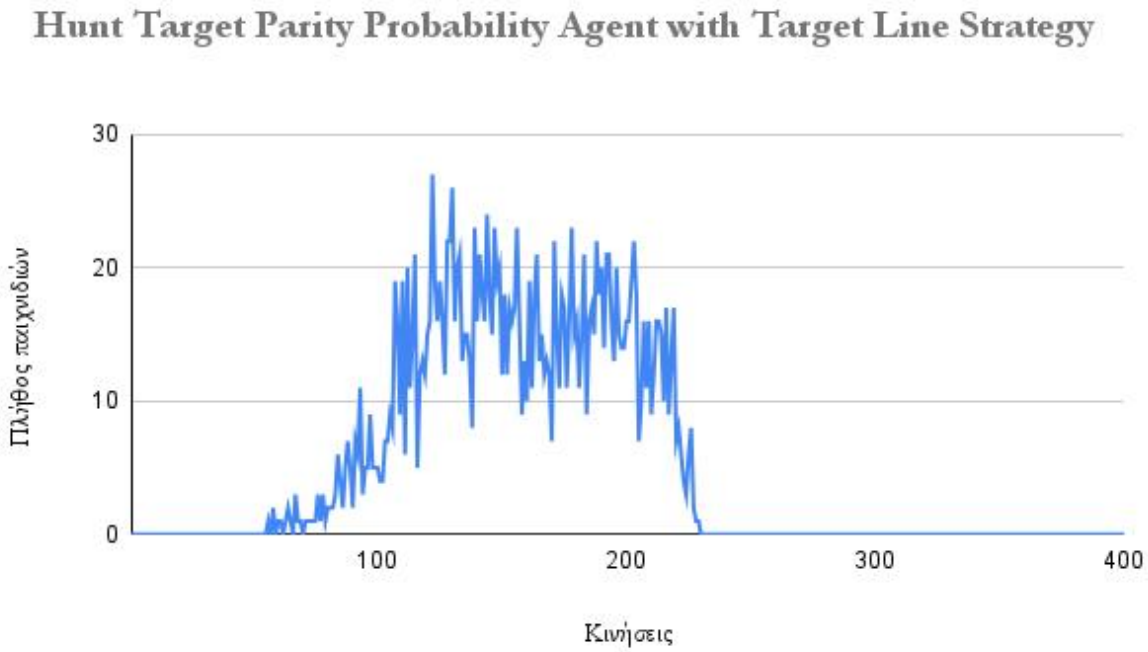
Σχήμα: 5-23



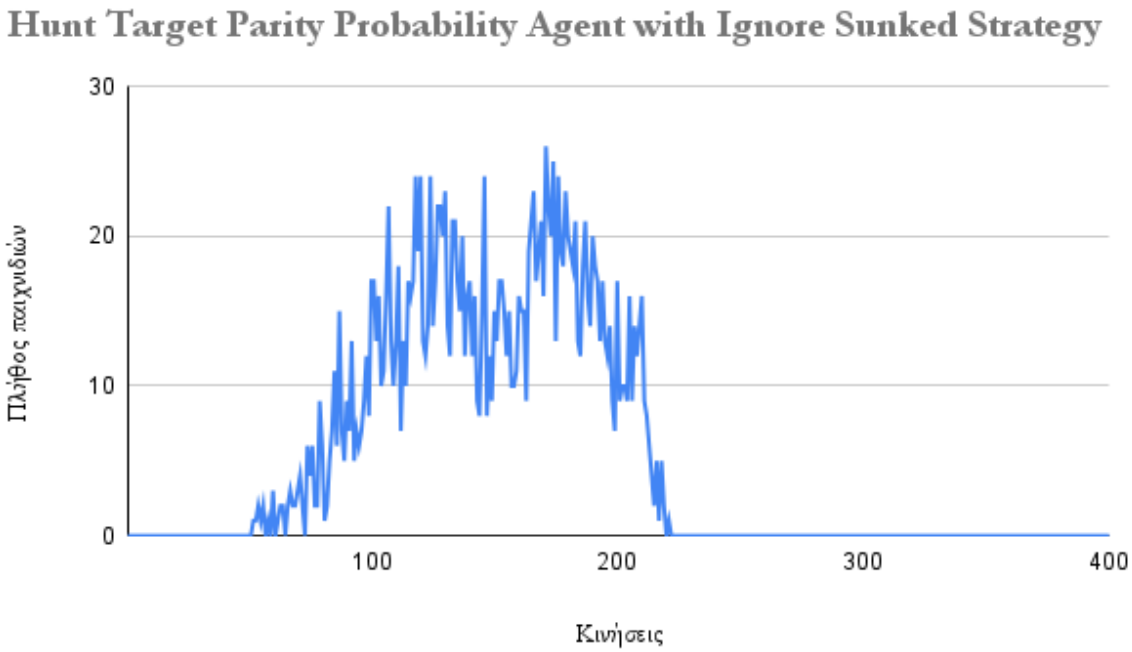
Σχήμα: 5-22



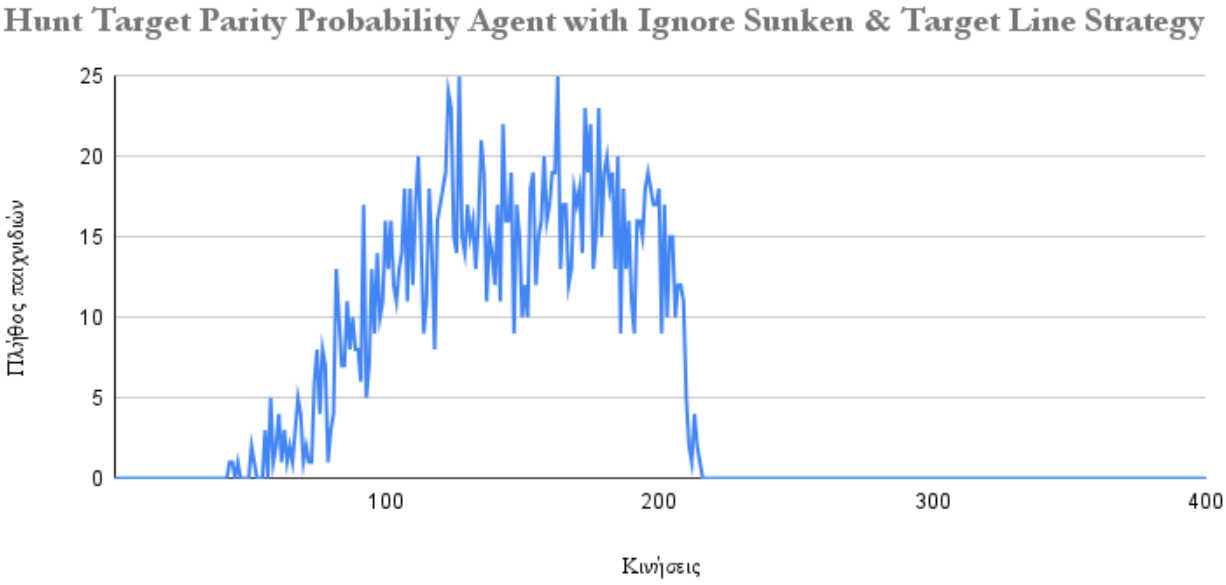
Σχήμα: 5-25



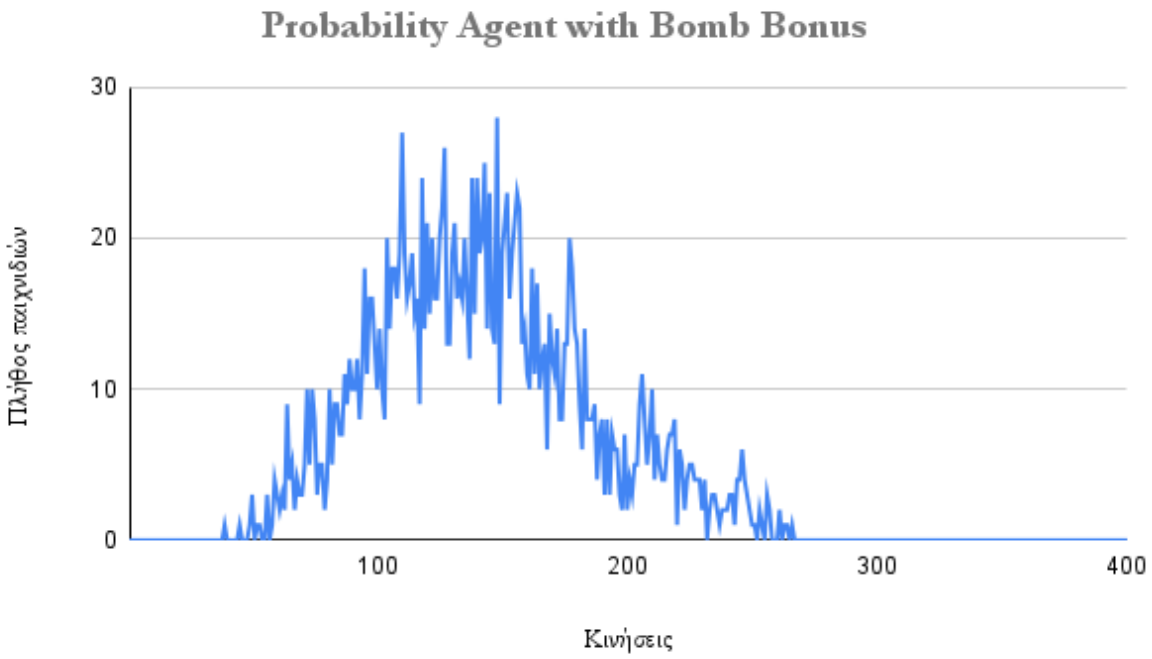
Σχήμα: 5-24



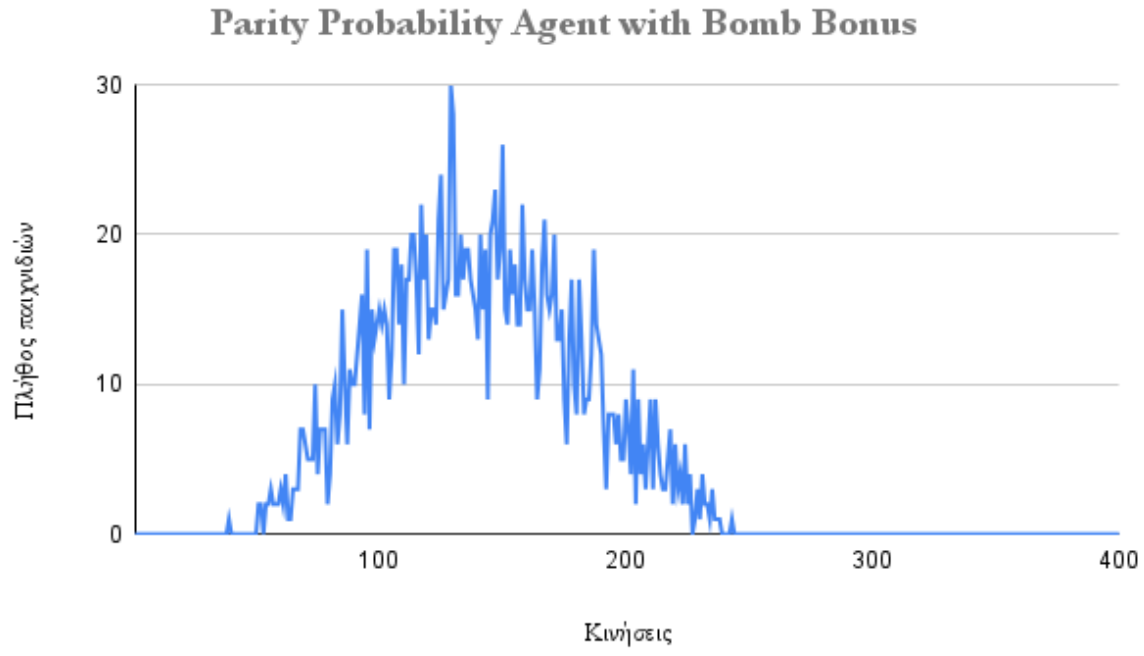
Σχήμα: 5-26



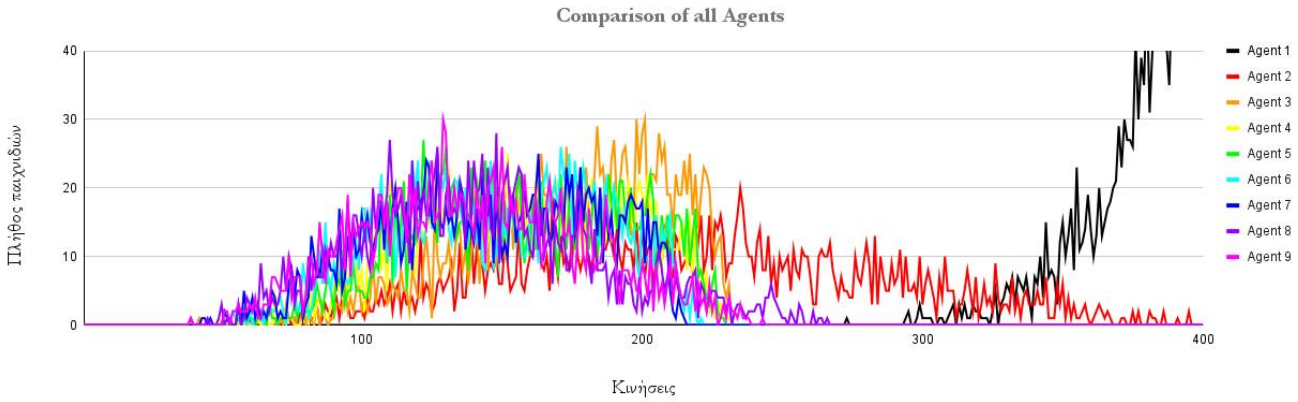
Σχήμα: 5-27



Σχήμα: 5-28



Σχήμα: 5-29



Σχήμα: 5-30

Παρακάτω δημιουργήσαμε έναν συγκεντρωτικό πίνακα με τις αποδόσεις και των 9 πρακτόρων.

	10x10 board		10x10 board με ATP στρατηγική		20x20 board	
Στατιστικά / Πράκτορες	Μέσος όρος κινήσεων	Τυπική απόκλιση	Μέσος όρος κινήσεων	Τυπική απόκλιση	Μέσος όρος κινήσεων	Τυπική απόκλιση
Agent 1	95,41	4,64	95,42	4,84	378,80	19,84
Agent 2	66,09	12,46	74,07	9,78	218,68	65,58
Agent 3	61,21	8,28	67,90	5,72	175,08	35,19
Agent 4	56,81	8,00	62,30	5,01	158,66	36,47
Agent 5	56,29	8,49	61,68	5,34	157,42	37,63
Agent 6	49,34	8,39	51,17	6,78	148,30	37,59
Agent 7	45,65	8,68	44,55	7,43	145,65	38,05
Agent 8	44,84	9,18	44,53	8,38	142,56	42,75
Agent 9	**44,51**	8,73	**44,07**	7,74	**140,39**	39,33

Για την περίπτωση 20x20 board, βλέπουμε ότι οι πράκτορες μας ακολούθησαν μία παρόμοια πορεία συγκριτικά με την περίπτωση 10x10 board. Μάλιστα, αναλογικά, οι πράκτορες μας απέδωσαν καλύτερα. Ο πράκτορας 9 έχει 44,51 κινήσεις κατά μέσο όρο σε παιχνίδι 100 κελιών, ενώ ο αντίστοιχος πράκτορας σε παιχνίδι 20x20 με 400 κελιά έχει 140 κινήσεις, ενώ θα περιμέναμε $44{,}51 * 400 / 100 = 178{,}04$ κινήσεις. Αυτό ισχύει για όλους τους πράκτορες, εκτός του τυχαίου, ο οποίος είχε κάκιστη απόδοση με 378 κινήσεις κατά μέσο όρο, αναλογικά παρόμοια απόδοση με την περίπτωση 10x10. Επίσης, παρόμοια αναλογικά ήταν και η (μικρή) βελτίωση με την εισαγωγή της

στρατηγικής “Target Line” στον Agent 5. Επίσης, ενώ στην περίπτωση 10x10 η προσθήκη του Parity στον πράκτορα 9 είχε ελάχιστη διαφορά, στην περίπτωση 20x20 ο αντίστοιχος πράκτορας 9 είχε βελτίωση κατά 2 σχεδόν κινήσεις κατά μέσο όρο, κάτι που μας δείχνει ότι στα περισσότερα κελιά, η προσθήκη του Parity στον συγκεκριμένο πράκτορα έχει κάποια σημασία. Τέλος, παρατηρούμε την μεγάλη διαφορά στην τυπική απόκλιση στην περίπτωση του 20x20 και του 10x10 board. Ήταν κάτι αναμενόμενο, καθώς το board τετραπλασιάστηκε σε μέγεθος, και έτσι αυξήθηκε και η αβεβαιότητα για το πού μπορεί να βρίσκεται ένα πλοίο. Έτσι, οι πράκτορες μας χρειάζονται περισσότερες κινήσεις για να βρουν τα πλοία και να λήξουν το παιχνίδι.

5.1 Σύγκριση αποτελεσμάτων με άλλες εργασίες

Στην υποενότητα αυτή θα συγκρίνουμε τα αποτελέσματα της εργασίας μας, τα οποία παρουσιάσαμε αναλυτικά προηγουμένως, με τα αποτελέσματα άλλων εργασιών. Στη συνέχεια, θα βγάλουμε τα δικά μας μικρά συμπεράσματα ως προς τι έγινε καλύτερο στις άλλες εργασίες και τι όχι.

Ο Sahoo, R. (2014) χρησιμοποιεί Monte Carlo sampling για να δημιουργήσει έναν δυνατό αλγόριθμο για την λύση του παιχνιδιού της Ναυμαχίας (10x10 board, 5 πλοία). Συγκεκριμένα, ο αλγόριθμος δειγματοληπτεί την belief state των τοποθεσιών των εχθρικών πλοίων N φορές. Έπειτα, επιλέγει να επιτεθεί στην άδεια τοποθεσία, η οποία περιέχει ένα σημείο πλοίου που δεν έχει χτυπηθεί ακόμα στο μεγαλύτερο μέρος των δειγμάτων αυτών. Για N=10 και 100 trials, ο μέσος όρος βολών για το Monte Carlo agent είναι 90,17 όταν έπαιξε ενάντια σε rule-based agent και 87,96 όταν έπαιξε ενάντια σε random agent. Οι ελάχιστες κινήσεις που χρειάστηκε κάποιο παιχνίδι για να λήξει ήταν 74 κινήσεις ενάντια στον rule-based και 75 ενάντια στον random, ενώ οι μεγαλύτερες ήταν 98 και 96 αντίστοιχα. Επίσης, ο rule-based agent έκανε 88 κινήσεις κατά μέσο όρο ενάντια στον Monte Carlo agent και ο random agent έκανε 90,5 κινήσεις. Βλέπουμε λοιπόν ότι ο συγκεκριμένος τρόπος δεν αποδίδει όπως αποδίδει η δική μας προσέγγιση. Βέβαια, δεν έχουμε βάλει τον δικό μας agent να παίξει εναντίον κάποιου άλλου, αλλά δεδομένου ότι δεν παίζει ιδιαίτερο ρόλο μόλις ξεκινήσει το παιχνίδι, μπορούμε να βγάλουμε ένα ασφαλές συμπέρασμα για την απόδοση της Monte Carlo προσέγγισης. Για N=100 και 100 trials, τα πράγματα γίνονται λίγο πιο ενδιαφέροντα. Ο Monte Carlo agent έχει τώρα 84,92 και 81,67 κινήσεις αντίστοιχα όταν παίζει με τους rule-based και random agents, με ελάχιστες κινήσεις για να λήξει το παιχνίδι στις 57 και 63 αντίστοιχα. Για τις μέγιστες κινήσεις, έχουμε 100 και 99 αντίστοιχα. Εδώ αποδίδει κάπως καλύτερα, αλλά ακόμα απέχει από την δική μας προσέγγιση, η οποία στην βέλτιστη μορφή της, είχε 44 κινήσεις κατά μέσο όρο.

Στην εργασία του Clementis, L. (2014), ο συγγραφέας προσεγγίζει το πρόβλημα της Ναυμαχίας με νευρωνικά δίκτυα, με δύο διαφορετικούς τρόπους, όπως έχουμε αναφέρει και στο Κεφάλαιο 2. Έπειτα, παίζει και μία ομάδα ανθρώπων το παιχνίδι. Με τον πρώτο τρόπο, χρησιμοποιώντας ένα νευρωνικό δίκτυο 3 επιπέδων (layers) μαζί με την μέθοδο Gradient Descent, και έχοντας δύο πλοία σε σχήμα Γ με διαστάσεις 2x3 σε ένα board 7x7, ο μέσος όρος κινήσεων για να βρει τα δύο πλοία είναι 19,85 κινήσεις. Στον δεύτερο τρόπο, χρησιμοποιώντας ένα νευρωνικό δίκτυο 3 επιπέδων πάλι, αλλά αυτή τη φορά μαζί με ενισχυτική μάθηση και με τα ίδια δεδομένα board και πλοίων, ο μέσος όρος κινήσεων είναι στις 20,87. Στη συνέχεια, ο τυχαίος πράκτορας, χρησιμοποιώντας ξανά νευρωνικά δίκτυα δίνει έναν μέσο όρο της τάξης των 44,37 κινήσεων. Τέλος, ο ανθρώπινος παίκτης, παίζοντας κανονικά, βρίσκει τα δύο πλοία σε σχήμα Γ σε 21 κινήσεις περίπου κατά μέσο όρο. Εδώ οφείλουμε να σημειώσουμε πως δεν μπορεί να γίνει σύγκριση, άμεση ή έμμεση, με τα δικά μας αποτελέσματα, καθώς στην εργασία αυτή χρησιμοποιούνται διαφορετικά σχήματα πλοίων, τα οποία καλύπτουν και ένα διαφορετικό συνολικό πλήθος θέσεων στο board, και ένα διαφορετικό μέγεθος board, 7x7 αντί για 10x10.

Τέλος, η εργασία από την General Atomics κάνει χρήση του Q-learning για να προπονήσει τον agent με δεδομένα που σύλλεξε η General Atomics από ανθρώπινους παίκτες, ώστε να τον βελτιώσει για να μπορεί να παίζει και να νικάει ανθρώπινους παίκτες. Στο τέλος του learning, ο agent κατάφερε να λήγει τα παιχνίδια σε 52 κινήσεις κατά μέσο όρο. Η δική μας βέλτιστη προσέγγιση έχει 44 κινήσεις κατά μέσο όρο στο ίδιο μέγεθος board και πλήθος στόλου, και έτσι βλέπουμε ότι η προσέγγιση μας είναι καλύτερη.

6 Συμπεράσματα

6.1 Σχολιασμός

Γίνεται σαφές λοιπόν ότι όλοι οι πράκτορες μας, εκτός του τυχαίου, έχουν καλές στρατηγικές, που μπορούν να εφαρμοστούν και σε περιπτώσεις με μεγαλύτερο board, αποδίδοντας καλύτερα από τα αναμενόμενα. Επίσης, καθοριστικό ρόλο έπαιξε η προσθήκη του μαθηματικού όρου Parity στην υλοποίησή μας, μειώνοντας στο μισό τα πιθανά κελιά που θα έπρεπε να αναζητήσουμε πλοίο, αλλά και η προσθήκη των πιθανοτήτων, δίνοντας μας τελικώς έναν πράκτορα βασισμένο εντελώς σε αυτές. Υπολογίζοντας σε ποιο κελί πιθανών να βρίσκεται κάθε φορά κάποιο πλοίο, και μετρώντας πόσες φορές μπορεί να περάσει κάποιο πλοίο από κάθε κελί, καταφέραμε να βελτιώσουμε πολύ το κομμάτι "Hunt" και να φτάσουμε τελικά σε έναν πράκτορα που μπορεί κατά μέσο όρο να ολοκληρώσει ένα παιχνίδι της Ναυμαχίας σε 44 μόλις κινήσεις, ή σε 140 αν παίζουμε την παραλλαγή του 20x20 board. Οι πράκτορες μας μπορούν να εφαρμοστούν σε οποιαδήποτε παραλλαγή board ή πλοίων και αν επιλέξει κάποιος.

6.2 Μελλοντικές προσθήκες

Φυσικά, η υλοποίηση μας δεν είναι τέλεια και έχει αρκετό χώρο για να δημιουργήσει κάποιος μελλοντικά διάφορες χρήσιμες προσθήκες. Μία από αυτές είναι η έξυπνη, real-time αλλαγή του Parity, ώστε κάθε φορά, αναλόγως ποιο πλοίο ψάχνει, να μεταβάλει το Parity με τέτοιο τρόπο ώστε να χτυπάει τα κατάλληλα κελιά. Για παράδειγμα, αν ψάχνουμε το Aircraft carrier με μέγεθος 5 θέσεων, τότε ο αλγόριθμος να χτυπήσει κάποια από τις πρώτες 5 θέσεις (1-5) και ύστερα η δεύτερη βολή του να είναι +5 θέσεις από την αρχή. Αν χτυπήσει την $3^{η}$ θέση, η επόμενη βολή του να είναι στην ίδια σειρά η $8^{η}$ θέση. Έτσι, με τις 2 αυτές βολές, αποκλείσαμε το ενδεχόμενο το Aircraft carrier να κρύβεται σε αυτή τη σειρά. Επιπλέον, καθώς η υλοποίηση μας δεν έχει καθόλου machine learning, και έτσι είναι ευάλωτη σε αντιπάλους που διαθέτουν, θα μπορούσε μία πιθανή προσθήκη να περιλαμβάνει μάθηση. Τέλος, κάποιος θα μπορούσε να φτιάξει στρατηγικές για πλοία με διαφορετικό σχήμα από τα σχήματα της παρούσας εργασίας, τα οποία είναι σε ευθεία γραμμή, όπως πλοία σε σχήμα Γ, τετράγωνα πλοία, κ.ο.κ

6.3 Τι μάθαμε

Η προσωπική μου ενασχόληση με την παρούσα εργασία μου έδειξε πόσο δύσκολο είναι τελικά για τον άνθρωπο να παίξει αποδοτικά κάποιο παιχνίδι, είτε αυτό είναι ντετερμινιστικό, είτε όχι. Στο board 10x10 τα πράγματα είναι πιο απλά, αλλά όσο αυξάνουμε το μέγεθος του board, οι αποφάσεις για τον άνθρωπο γίνονται δύσκολες πολύ γρήγορα. Για ένα board 50x50, με 2500 κελιά και ίδιο αριθμό και μέγεθος πλοίων, όπως και στην περίπτωση του 10x10, μπορεί να καταλάβει κάνεις εύκολα πόσες επιλογές έχει κάποιος για να ρίξει τη πρώτη βολή. Δεδομένου άπλετου χρόνου, ο άνθρωπος υποθετικά θα μπορούσε να παίζει το ίδιο αποδοτικά με τον υπολογιστή, αλλά αυτό δεν είναι παρά ένα τελείως υποθετικό σενάριο, καθώς σχεδόν ποτέ δεν υπάρχει άπλετος (= πολύς) χρόνος για να κάνει την επόμενη κίνηση. Έτσι, δεν μου κάνει καμία εντύπωση που ακόμα και Grand Masters στο σκάκι, όπως ο Garry Kasparov, τελικά έχασαν από τον υπολογιστή. Η δυνατότητα αυτή που έχουμε, έχοντας δημιουργήσει τέτοιες μηχανές με τέτοιες δυνατότητες, θα μπορούσε στο μέλλον να μας φανεί απίστευτα χρήσιμη, αν ποτέ συμβεί κάποια επανάσταση των μηχανών, καθώς μόνο μία μηχανή θα μπορεί να νικήσει κάποια άλλη μηχανή.

Βιβλιογραφικές αναφορές

Sutton, R.S., Barto, A.G. (1998), "Reinforcement Learning: An Introduction". *MIT Press*, Cambridge, MA, 1998. Second Edition, *MIT Press*, 2018.

Tesauro, G. (1995), "Temporal Difference Learning and TD-Gammon". *Communications of the ACM*, vol. 38, Issue 3, pp. 58-68, March 1995.

Sevenster, M. (2004), "Battleships as a decision problem". *ICGA Journal*, 27, 3, pp. 142–149

Bridon, J.G., Correll, Z.A., Dubler, C.R. & Gotsch Z.K. (2009), "An artificially intelligent Battleship player utilizing adaptive firing and placement strategies". Διαθέσιμο στο www.cores2.com/files/FinalResearchPaper.pdf.

Dellaert, F., Fox, D., Burgard, W., Thrun, S. (1999), "Monte Carlo Localization for Mobile Robots". *IEEE International Conference on Robotics and Automation* (ICRA99). May, 1999.

Metropolis, N., Ulam, S., "The Monte Carlo Method". *Journal of the American Statistical Association*, vol. 44, pp. 335-341, Sept. 1949

Monniaux, D. (2001). An Abstract Monte-Carlo Method for the Analysis of Probabilistic Programs. *ACM SIGPLAN Notices*, vol 36, pp. 93-101.

Banzhaf, W., Koza, J. R., Ryan, C., Spector, L., and Jacob, C., "Genetic programming," *in IEEE Intelligent Systems and their Applications*, vol. 15, no. 3, pp. 74-84, May-June 2000

Benson, K. (2000), "Evolving Finite State Machines with Embedded Genetic Programming for Automatic Target Detection". *Congress on Evolutionary Computation*, vol. 2, pp. 1543-1549.

Mysinger, M. (1998), "Genetic Design of an Artificial Intelligence to Play the Classic Game of Battleship". *Genetic algorithms and genetic programming at Stanford*, Stanford: California

Roy, A. (2000), "Artificial Neural Networks - A Science in Trouble". *2000 ACM SIGKDD*, vol. 1, n. 2, pp. 33-38.

Sahoo, R., (2014), "Battleships – A Game Playing Agent". Διαθέσιμο στο https://www.scribd.com/document/207585036/Cs6601-Project-1-Proposal. Πλήρες κείμενο διαθέσιμο στο https://www.dropbox.com/s/ek13fmp6iwz06hs/cs6601_project_1_paper.pdf?dl=0.

Clementis, L. (2014), "Study of Game Strategy Emergence by using Neural Networks". Διαθέσιμο στο http://acmbulletin.fiit.stuba.sk/abstracts/clementis2014.pdf.

General Atomics, GA-CCRi, "Deep Reinforcement Learning–of how to win at Battleship". Διαθέσιμο στο https://www.ga-ccri.com/deep-reinforcement-learning-win-battleship

Γουργιώτης, Γ., (2011), "Δημιουργία Πράκτορα Τεχνητής Νοημοσύνης για Παιχνίδι Στρατηγικής". Διπλωματική εργασία που έγινε στο Πανεπιστήμιο Πατρών. Διαθέσιμη στο https://nemertes.library.upatras.gr/jspui/bitstream/10889/4989/3/Nimertis_Gourgioti.pdf