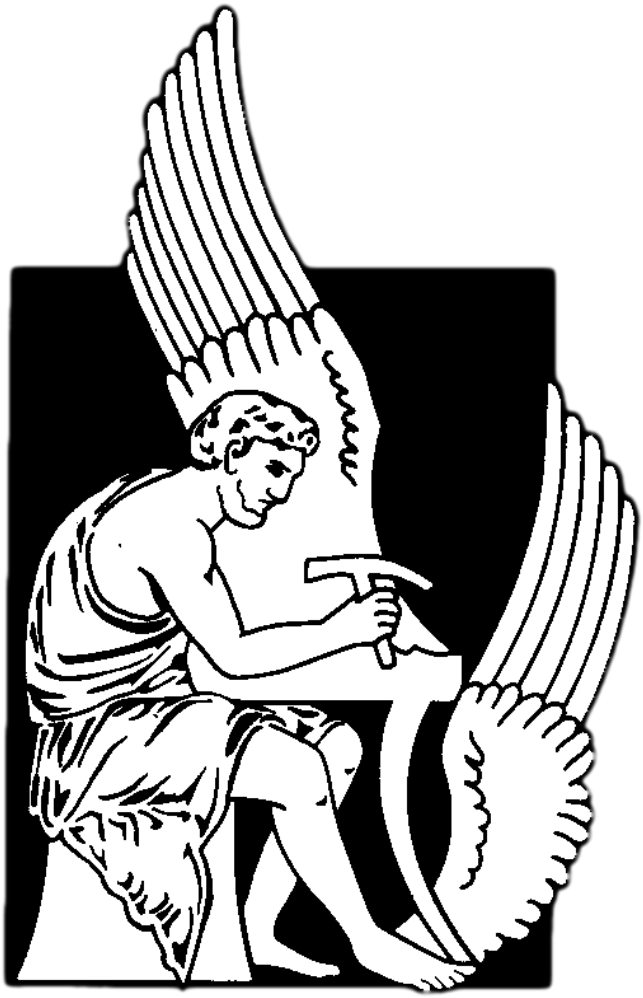
ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
ΤΜΗΜΑ ΗΛΕΚΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΠΛΗ517 «ΠΟΛΥΠΡΑΚΤΟΡΙΚΑ ΣΥΣΤΗΜΑΤΑ»



Υλοποίηση Πράκτορα για το Pandemic Board Game

2014030064	**Γερωνυμάκης Παύλος**
2016030023	**Αποστολόπουλος Αθανάσιος-Ιάκωβος**
2016030125	**Νίκου Γεώργιος Νεκτάριος**

Περιγραφή αναφοράς

Σε αυτή την αναφορά παρουσιάζεται η μεθοδολογία που ακολουθήθηκε για την υλοποίηση της εργασίας. Ο τρόπος σκέψης, η λήψη αποφάσεων και ο τρόπος εκτέλεσης αναφέρεται παρακάτω με την σειρά με την οποία διαδραματίστηκε κατά τις συναντήσεις της ομάδας και στον προσωπικό χρόνο του κάθε φοιτητή.

Πρώιμο στάδιο

Στην πρώτη φάση για να επεκτείνουμε την κατανόηση μας πάνω στους κανόνες του παιχνιδιού και στους ρόλους των παικτών, αναζητήσαμε ιδανικές στρατηγικές σε συμβουλευτικά forums [1][2] και σε recorded gameplay[3].Με βάση την επιλεγμένη πληροφορία που αποκομίσαμε, δημιουργήσαμε στρατηγικές για κάθε ρόλο και μια στρατηγική γενικού σκοπού. Οι στρατηγικές βασίζονται στις ιδιαιτερότητες του κάθε ρόλου και εξυπηρετούν την εκπλήρωση τον ατομικών τους στόχων. Η γενική περίπτωση βασίζεται στην πεποίθηση καθαρισμού του χάρτη και την αντιμετώπιση εκτάκτων αναγκών. Το θεωρητικό κομμάτι του σχεδιασμού προέκυψε με καταιγισμό ιδεών από την ομάδα και η υλοποίηση με ευριστικούς κώδικες έγινε για κάθε ρόλο ξεχωριστά.

General player: Για κάθε non cured ασθένεια που μαζεύει >1 κάρτες τις αποθηκεύει σε μια μεταβλητή ώστε να αποφασίσει αν θα μείνουν ανέγγιχτες ή αν θα χρησιμοποιηθούν για κάποια μετακίνηση/κτίσιμο RS. Εφόσον μαζέψει 4 κάρτες (3 για Scientist) ίδιου χρώματος ο agent προσπαθεί να κινηθεί προς RS για να πραγματοποιήσει το cure της αντίστοιχης ασθένειας.

Operations Expert: Ο προσωπικός στόχος του OPEX είναι μέχρι το τέλος του δεύτερου γύρου να έχει χτίσει RS (Research Stations) σε όλες τις ιδανικές πόλεις.

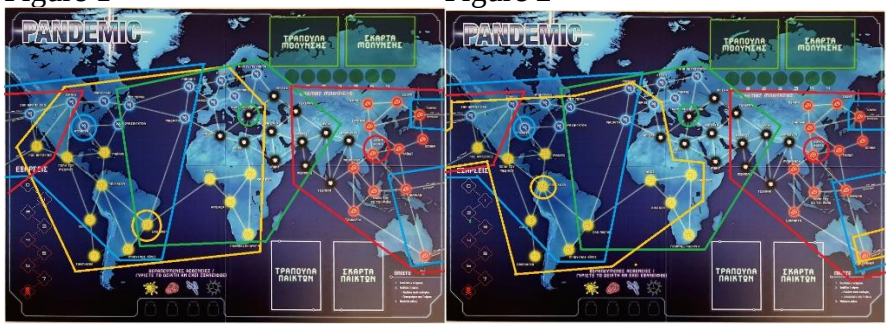
Ο ορισμός των ιδανικών πόλεων OCRS(Optimal Cities for Research Stations) προέκυψε μέσα από αρχική μας έρευνα. Αρχικά είχαμε δύο μοντέλα όπου το πρώτο περιελάμβανε έξι OCRS(Figure 3-appendix)

- Atlanta Paris Bogota Khartoum Karachi Hong Kong

ενώ το δεύτερο περιλαμβάνει τέσσερα OCRS.

- Atlanta, Bogota, Istanbul, Hong Kong (Figure 1)
- Atlanta, Sao Paulo, Istanbul, Hong Kong (Figure 2)

Figure 1 Figure 2



Τελικά αποφασίστηκε ότι θα χρησιμοποιηθεί το μοντέλο τον τεσσάρων OCRS για να αφήνει τα εναπομείναντα δύο RS για ελεύθερη κατανομή. Οι περιοχές που φράσσονται από τα σύνορα του κάθε RS στα Figures 1,2 μαρκάρουν την μέγιστη εμβέλεια MR(Maximum Range). Ως MR μίας OCRS έχουμε ορίσει την απόσταση τρία

από την OCRS για να καλύψουμε την περίπτωση της μετακίνησης ενός παίκτη προς την OCRS και την εκτέλεση Cure μέσα σε ένα γύρο. Έτσι όλα τα σημεία στο χάρτη βρίσκονται μέσα στην εμβέλεια ενός OCRS γλιτώνοντας από του παίκτες έξτρα μετακινήσεις.

Αφού ορίσαμε τις OCRS κατασκευάζουμε μία στρατηγική για τον OPEX με την οποία καταλήγει στο τέλος του δεύτερου γύρου να έχει τοποθετήσει RS σε όλες τις OCRS.
Συγκεκριμένα ο παίκτης θα έχει χτίσει RS στις πόλεις Bogota, Istanbul, Hong Kong. Από τον τρίτο γύρο και μέχρι το τέλος θα ακολουθεί την γενική στρατηγική.

Quarantine Specialist: Για τον συγκεκριμένο ρόλο ακολουθήσαμε μια στρατηγική, η οποία έδινε έμφαση στην μετακίνηση στις απομακρυσμένες περιοχές με τους περισσότερους κύβους. Ο QS δεν χρειάζεται να κάνει απαραίτητα treat στον γύρο του, αρκεί να μεταβεί σε μια πόλη που κινδυνεύει από outbreak. Με τη special ability του μπορεί να αποτρέψει την δημιουργία outbreak με την ύπαρξη του σε αυτήν την πόλη ή σε μία γειτονική.
Πιο συγκεκριμένα ο QS ιεραρχεί τις περιοχές που θα επισκεφθεί με βάση πρώτα τον αριθμό των κύβων και δευτερευόντως με βάση την απόσταση από την πόλη που βρίσκεται. Αν ο QS βρίσκεται σε πόλη με 2 ή 3 κυβάκια αφαιρεί ένα και έπειτα κάνει αναζήτηση σε radius remainingActions. Για παράδειγμα μια πόλη με 3 κύβους και απόσταση 4 προτιμάται από μια πόλη με 3 κύβους και απόσταση 3. Επιπλέον στην ισοψηφία πόλεων σε απόσταση και κύβους, προτιμάται η πόλη με τους περισσότερους γείτονες για αποφυγή διάδοσης σε περίπτωση Outbreak.

Scientist: Η στρατηγική μας βασίζεται στην παραδοχή ότι ο Scientist είναι ο πιο efficient ρόλος για να κάνει οποιοδήποτε cure. Όπως κάθε ρόλος για κάθε non cured ασθένεια που μαζεύει >1 κάρτες τις αποθηκεύει σε μια μεταβλητή ώστε να μπορεί να κρίνει αν αυτές θα χρησιμοποιηθούν και επιπλέον φροντίζει να βρίσκεται πάντα σε απόσταση <3 από οποιοδήποτε Research Station, ώστε να μπορεί να πραγματοποιεί άμεσα cure σε ένα γύρο μόλις αυτό είναι εφικτό. Δευτερευόντως, αν δεν μπορεί να κάνει cure, προσπαθεί να βρει πόλεις με κύβους κοντά σε RS για να κάνει treat.

Medic: Ο στόχος του Medic είναι σε κάθε γύρο να κάνει treat σε πόλεις με τα περισσότερα κυβάκια ώστε να αξιοποιείται στο μέγιστο η ικανότητα του ρόλου. Ειδικότερα ο Medic κάνει αναζήτηση για την πόλη με τα περισσότερα κυβάκια σε radius (remainingActions - 1) ώστε να βρει την πόλη με τους περισσότερους συνολικά κύβους, μεταβαίνει σε αυτή και κάνει treat. Η διαδικασία αυτή επαναλαμβάνεται από την αρχή μέχρι να τελειώσουν οι διαθέσιμες κινήσεις του γύρου.

Evaluation Function πρώιμου σταδίου
Για την αξιολόγηση των προσομοιώσεων του MCTS αναπτύξαμε ένα δικό μας evaluation function έχοντας ως βάση τις παραμέτρους και τις μεταβλητές του παιχνιδιού. Η βαθμολόγηση γίνεται ως εξής:

$$u(board) = -100 \cdot AD - \sum_{city\ i} c_i - \sum_{\substack{city\ i \\ where\ ci=3\ and\ !QS}} c_i \cdot n_i - 10 \cdot ob + 5 \cdot RS + 5 \cdot oRS + \sum_{\substack{city\ i \\ wi<4}} \frac{4 - w_i}{4} \cdot c_i + 10 \cdot \sum_{card\ i} cd_i$$

,όπου AD ο αριθμός των Active Diseases, c_i ο αριθμός των κύβων σε μία πόλη i, n_i ο αριθμός των γειτόνων μίας πόλης i, ob ο αριθμός των outbreak που έχουν συμβεί έως τώρα, RS ο αριθμός των κτισμένων Research Station, oRS ο αριθμός των Optimal RS, w_i η ελάχιστη απόσταση μίας πόλης από κάποιον παίκτη, και cd_i ο μέγιστος αριθμός των καρτών ενός μη cured χρώματος που έχει ένας παίκτης. Οι τιμές πήραν αυτή τη μορφή γιατί θέλαμε να δώσουμε μεγαλύτερο κίνητρο στους παίκτες με σειρά προτεραιότητας: να πραγματοποιούν cures, να κάνουν treat σε επικίνδυνες πόλεις(max cubes = 3 και πολλούς γείτονες) και κατ' επέκταση να αποφεύγουμε τα outbreaks, να χτίζουν RS σε optimal περιοχές, και ως τελευταία επιλογή να εστιάζουν στις πόλεις με λιγότερα από 3 κυβάκια.

Monte Carlo Tree Search Model

Ο αλγόριθμος Monte Carlo Tree Search αποτελεί έναν πιθανολογικό αλγόριθμο αναζήτησης δέντρου. Συνδυάζει την κλασική αναζήτηση δέντρου με την μέθοδο Monte Carlo και είναι ευρέως διαδεδομένος στην λύση παιχνιδιών με μεγάλο branching factor και άρα δύσκολο να λυθούν με brute force. Ένας άλλος υποψήφιος αλγόριθμος που σκεφτήκαμε να υλοποιήσουμε είναι ο Minimax σε συνδυασμό με κάποια επέκταση του (π.χ. a,b pruning). Προτιμήθηκε, τελικά, μια παραλλαγή του MCTS λόγω της καλής απόδοσης του σε δέντρα με μεγάλο βάθος και αρκετά παιδιά και της έλλειψης εμπιστοσύνης στην evaluation function που αναπτύξαμε για την αποτελεσματική λειτουργία του minimax.

Ο αλγόριθμος MCTS αποτελείται από 4 στάδια:

- Selection
- Expansion
- Simulation
- Backpropagation

Αρχικά στο στάδιο του Selection ο αλγόριθμος ξεκινάει στην ρίζα διαλέγοντας το πιο υποσχόμενο παιδί με βάση το UCT μέχρι να βρεθεί σε φύλλο. Στο Expansion επεκτείνει το δέντρο μέχρι να καλυφθούν όλες οι επιθυμητές καταστάσεις. Έπειτα στο Simulation παίζεται αυθαίρετα το παιχνίδι μέχρι να φτάσει σε τελική κατάσταση. Τέλος στο Backpropagation ο αλγόριθμος αξιολογεί την τελική κατάσταση και διασχίζει προς τα πίσω το δέντρο για να ενημερώσει τον αριθμό των επισκέψεων σε κάθε κόμβο που επισκέφθηκε καθώς και το σκορ του.Αυτή η διαδικασία των τεσσάρων σταδίων επαναλαμβάνεται μέχρι ένα καθορισμένο όριο επαναλήψεων.

<u>Υλοποίηση</u>

Σε πρώτη φάση υλοποιήσαμε το MCTS με την παραπάνω evaluation function την οποία αξιοποιήσαμε στις ευριστικές αναζητήσεις. Καταλήξαμε, ύστερα και από συνάντηση με τον κ. Βογιατζή, σε ένα μοντέλο δέντρου με 5 επίπεδα (4 επίπεδα action + 1 ρίζα) το οποίο καλύπτει το παιχνίδι μέχρι να ξανά έρθει η σειρά του να παίξει. Επιπλέον γίνεται Simulation μέχρι να ξαναέρθει η σειρά του παίκτη. Προκειμένου να το simulation να είναι όντως random, ανακατεύουμε όλα τα decks,

πριν την εκτέλεση. Το simulation πραγματοποιείται στους τελικούς κόμβους του δένδρου που δημιουργείται γιατί αναζητούμε 4αδες από actions και όχι τα καλύτερα μεμονωμένα actions σε κάθε επίπεδο. Οπότε το sequence που επιστρέφεται προκύπτει από το καλύτερο κόμβο που βρίσκεται στο επίπεδο 4, και στη συνέχεια αναζητούμε τους προγόνους του.
Το UCT περιγράφεται ώς εξής:

$$UCT = \frac{valuation}{nodeVisits} + \sqrt{2} * \frac{\log totalVisits}{nodeVisit}$$

Σε δεύτερη φάση επεκτείναμε το βάθος του δέντρου φθάνοντας τώρα σε βάθος number of player + 1 * 4 + 1 ρίζα(25 για 4 παίκτες) , καθώς και το εύρος της προσομοίωσης μέχρι το τέλος του παιχνιδιού για να επιτύχουμε καλύτερα αποτελέσματα.

Επίσης αναπτύξαμε ένα νέο evaluation function για να αξιολογούμε την τελική κατάσταση του παιχνιδιού, και συγκεκριμένα αν νικήσαμε ή στο πόσο κοντά ήμασταν στο να νικήσουμε, και έτσι επιστρέφει τις εξής τιμές:

- +1 σε περίπτωση νίκης
- 0 σε περίπτωση ήττας από κάρτες
- -1 σε κάθε άλλη περίπτωση ήττας
- +0.25 για κάθε cure(σε περίπτωση ήττας)

Κατά το στάδιο του expansion, δημιουργούμε τα παιδιά του κάθε κόμβου καλώντας τη συνάρτηση generate nodes. Κατά τη συνάρτηση αυτή εξετάζουμε ποιες κινήσεις επιτρέπονται να γίνουν σε αυτό το σημείο του παιχνιδιού αλλά και ποιες έχουν νόημα να εξερευνηθούν/πραγματοποιηθούν. Επειδή είναι υπολογιστικά βαρύ να εξετάσουμε κάθε legal action που μπορεί να κάνει ένας agent, θα μπορούσαμε να έχουμε 48 παιδιά για charter flight, 6+ παιδιά για direct flight, το πολύ 6 drive/ferry(μέγιστος αριθμών γειτόνων στο χάρτη(Istanbul)), 4 παιδιά για ένα treat κάθε ασθένειας, και από ένα παιδί για τα actions cure και build RS. Σε ένα τέτοιο δέντρο είτε η εξερεύνηση actions θα καθίστανται ανεπαρκής εάν απαιτούσαμε λύση σε λογικά χρονικά πλαίσια είτε ο υπολογιστικός χώρος και χρόνος θα ήταν πάρα πολύ μεγάλος. Οπότε ύστερα από τη συζήτηση με τον κ. Βογιατζή αλλά και μεταξύ μας καταλήξαμε, στη δημιουργία ενός παιδιού για κάθε μορφή action, αν αυτή επιτρέπεται. Παρακάτω περιγράφεται πως δημιουργείται το κάθε παιδί.

Cure Disease Child

Ο agent εξετάζει αν τα χαρτιά του αρκούν για να πραγματοποιήσει κάποιο. Αν αυτό είναι δυνατό, ελέγχει αν βρίσκεται σε RS, οπότε και το κάνει, αν όχι αναζητεί το κοντινότερο RS και αρχίζει να πηγαίνει προς αυτόν.

Treat Disease Child

Ο agent επιλέγει κάνει treat το disease χρώματος ίδιου με το χρώμα του μέγιστου αριθμού κύβων ίδιου χρώματος.

Build Research Station Child

Αν ο αριθμός των ήδη κτισμένων RS είναι μικρότερος ή ίσος του 6, ο πράκτορας εξετάζει αν μπορεί να κτίσει σε αυτήν, δηλαδή αν δεν υπάρχει ήδη RS εκεί, έχει την κάρτα πόλης στην οποία «πατάει» στο χέρι του ή είναι OPEX.

Direct Flight Child

Ο agent εξετάζει ποια από τις πόλεις που αντιστοιχούν στις κάρτες του βρίσκεται στην πιο επικίνδυνη περιοχή, δηλαδή:

$$argmax\big(f(i)\big) = \{i | i \in C \ \cap i \in Cd\}$$

$$f(i) = \sum_{d=0}^{d=3} c_i$$

όπου ci ο αριθμός των κύβων σε μία πόλη i, που βρίσκονται σε απόσταση d από την πόλη που εξετάζουμε να πάμε.
Αντιμετωπίζει κάθε δυνατό προορισμό ως ένα κέντρο και υπολογίζει τον αριθμό των κύβων που βρίσκονται σε αυτή καθώς και σε πόλεις που βρίσκονται σε απόσταση μικρότερη του 4 από εκείνη.

Charter Flight Child

Αν μία από τις κάρτες που διαθέτει στο χέρι του ο πράκτορας αντιστοιχεί στην πόλη στην οποία βρίσκεται, εξετάζει σε όλον τον χάρτη ποια πόλη στην οποία δεν μπορεί να μεταβεί σε αυτόν τον γύρο χρειάζεται τη βοήθεια του, δηλαδή δεν βρίσκεται παίκτης κοντά και είναι κοντά σε outbreak ή πηγαίνει στην πόλη στην οποία μπορεί να αφαιρέσει τα περισσότερα κυβάκια χρώματος για τα οποία έχουμε λιγότερο από 6 διαθέσιμα.

Shuttle Flight Child

Κατά τη δημιουργία αυτού του παιδιού αναζητείται ένας RS(αν βρίσκεται ήδη σε RS) από τον οποίο μπορεί να μεταβεί στον ίδιο γύρο σε επικίνδυνη πόλη, δηλαδή στις κινήσεις που του απομένουν.

OPEX Flight Child

Η δημιουργία του OPEX flight child ακολουθεί την ίδια λογική με το charter flight, ελέγχοντας όμως αν βρίσκεται σε RS, αν είναι η πρώτη εκτέλεση μέσα στο γύρο και έχει τουλάχιστον μία κάρτα

Drive/Ferry Child

Ο πράκτορας μας αναζητεί την πιο επικίνδυνη πόλη(μέγιστος αριθμός κύβων ίδιου χρώματος και τουλάχιστον 1 κυβάκι) που βρίσκεται σε radius ίσο με τον αριθμό κινήσεων που του απομένουν μείον ένα(μία έξτρα κίνηση για treat για όλους τους παίκτες εκτός τον QS), και στην οποία δεν είναι τοποθετημένος κάποιος άλλος παίκτης. Αν μία τέτοια πόλη βρεθεί, πηγαίνει προς αυτήν, αν όχι μεγαλώνει το εύρος ώστε να μπορεί να μεταβεί σε αυτήν σε επόμενους γύρους.

Μοντελοποίηση συμπαικτών

Η στρατηγική που ακολουθεί ο πράκτορας μας, βασίζεται στην υπόθεση ότι οι συμπαίκτες του παίζουν βέλτιστα ή αρκετά κοντά σε αυτό. Αυτή η υπόθεση διευκολύνει τον πράκτορα στις αποφάσεις που πρέπει να πάρει για το γύρο του καθώς θεωρεί ότι οι συμπαίκτες του μπορούν να καλύψουν τις δύσβατες/απομακρυσμένες γι' αυτόν (αλλά κοντινές γι' αυτούς) περιοχές. Δε χρειάζεται, δηλαδή να ξοδεύει actions μετακινήσεων προκειμένου να μεταβεί σ' αυτές, αλλά ούτε να σπαταλάει χρόνο κατά την λήψη αποφάσεων αναζητώντας αν αυτές οι κινήσεις είναι αποδοτικές. Προσπαθεί δηλαδή πάντα να πετύχει το μεγαλύτερο τελικό evaluation, λειτουργώντας σαν ένας maximax agent. Η υπόθεση αυτή περιέχει το ρίσκο της παραμέλησης ασθενειών ή περιοχών που ο πράκτορας δε μπορεί εύκολα να φτάσει, όταν οι συμπαίκτες του δεν είναι βέλτιστοι(dummies).

Αντίθετα, μια minimax regret-like approach που δοκιμάσαμε μπορεί να φέρνει καλύτερα αποτελέσματα σε τέτοια παιχνίδια, όμως στα παιχνίδια με πράκτορες μπορούν να πάρουν ορθολογικές αποφάσεις, το ποσοστό νικών σε σχέση με την προηγούμενη προσέγγιση μειώνεται αρκετά. Αυτό συμβαίνει γιατί πολλές φορές δύο ή παραπάνω agents προσπαθούν να λύσουν το ίδιο υπό-πρόβλημα(συγκέντρωση σε μία μόνο μολυσμένη πόλη/περιοχή).

Suggestions

Το παιχνίδι επιτρέπει την επικοινωνία μεταξύ των παικτών με τη μορφή suggestions. Στην αρχή κάθε γύρου, κάθε παίκτης που δεν παίζει σε αυτόν τον γύρο έχει την επιλογή να στείλει μια πρόταση στον παίκτη που είναι η σειρά του(current player). Η πρόταση αυτή αποτελείται από το πολύ 4 κινήσεις, που ένας παίκτης έχει στη διάθεση του στο γύρο του. Οι 4 αυτές κινήσεις μπορούν να αποτελούν είτε τις κινήσεις που ο ίδιος ο παίκτης θα ακολουθούσε αν ήταν στη θέση του current player ή κινήσεις που θα βοηθήσουν αυτόν όταν έρθει η σειρά του.

Suggestion Construction

Το περιβάλλον στο οποίο λειτουργεί ο πράκτορας είναι στοχαστικό και ημι-δυναμικό. Προβλέψεις για επόμενες καταστάσεις, του παιχνιδιού είναι δύσκολες έως απίθανες. Για αυτόν τον λόγο και ο πράκτορας μας αντιμετωπίζει κάθε γύρο σαν ένα διακριτό παιχνίδι στο οποίο λαμβάνει αποφάσεις για τη στρατηγική που θα ακολουθήσει σε αυτόν τον γύρο και όχι μια πολιτική για ολόκληρο το παιχνίδι(με εξαίρεση τον τρόπο που αποταμιεύει τις κάρτες με ούτως ώστε να μπορεί να κάνει cure). Ομοίως, η προσπάθεια να καθοδηγήσουμε τους υπόλοιπους agents ώστε να πραγματοποιήσουν κινήσεις, που δε βασίζονται απλά στην κατάσταση του Board τη στιγμή που καλούνται να πάρουν την απόφαση τους, αλλά στοχεύουν στη διευκόλυνση της δικιάς μας στρατηγική θεωρείται από εμάς μάταιη και εγωιστική. Μάταιη γιατί για παράδειγμα σε παιχνίδι 4 παικτών η πρόταση του 4ου σε σειρά παίκτη στον πρώτο χαρακτηρίζεται ως μη αποδοτική αφού μέσα στους επόμενους τρεις γύρους που θα παιχτούν θα μολυνθούν τουλάχιστον 6 πόλεις, με πιθανά outbreaks που θα μπορούσαν να αποφευχθούν, αν αυτός είχε εστιάσει στο πρόβλημα που έχει μπροστά

του. Ακόμα και οι προτάσεις που οδηγούν σε κτίσιμο Research Station, προκειμένου ο παίκτης να προλάβει να ανακαλύψει κάποιο cure στο γύρο μπορούν να καταστούν αχρείαστες, αν κάποιος άλλος προλάβει να κάνει cure πριν από αυτόν. Εγωιστικές, γιατί οι προτάσεις και οι ενέργειες μπορεί να κατασκευάζονται με βάση ένα evaluation function που ανταμείβει τις «καλές» ενέργειες του πράκτορα και να αγνοεί την συνολική κατάσταση του παιχνιδιού. Οπότε οι προτάσεις που στέλνουμε στους συμπαίκτες μας, κατασκευάζονται με τον ίδιο τρόπο, που κατασκευάζουμε τα actions του πράκτορα στη σειρά μας, παίρνοντας υπόψη φυσικά τον ρόλο, τις κάρτες και τη θέση του στο χάρτη.

Suggestion Evaluation

Ο πράκτορας μας εκτός από την ικανότητα να στέλνει suggestions, πρέπει να είναι ικανός να **λαμβάνει τις προτάσεις των άλλων**, να τις **αξιολογεί**, να **αποφασίζει αν πρέπει να τις ακολουθήσει**, αλλά και να μπορεί να **αξιολογεί συνολικά τους συμπαίκτες του βάση αυτών**.

Για τη λήψη προτάσεων από τους συμπαίκτες μας χρησιμοποιούμε τις κατάλληλες συναρτήσεις του Board.

Αξιολογούμε τις προτάσεις των συμπαικτών μας με δύο τρόπους.

- Εκτελούμε τις προτάσεις των παικτών σε κάποιο αντίγραφο του Board και στη συνέχεια ελέγχουμε την κατάσταση του, χρησιμοποιώντας την evaluation function που αναπτύξαμε στο πρώιμο στάδιο(short-term evaluation). Χρησιμοποιούμε αυτή τη συνάρτηση γιατί τη θεωρούμε καταλληλότερη για να κρίνει την κατάσταση του Board μετά από μία 4αδα ενεργειών.

- Έπειτα έχοντας εκτελέσει τις 4αδες που προτείνονται δανειζόμαστε τη λογική που χρησιμοποιείται στον αλγόριθμο Monte Carlo και εκτελούμε επανειλημμένα random plays μέχρι το τέλος του παιχνιδιού. Στη συνέχεια ελέγχουμε την τελική κατάσταση του board κάνοντας χρήση της evaluation function, που αναπτύξαμε και χρησιμοποιούμε για το MCTS. Χρησιμοποιούμε αυτή τη συνάρτηση γιατί τη θεωρούμε καταλληλότερη για να μας δείξει αν οδηγούμαστε σε νίκη ή ήττα, και πόσο κοντά ήμασταν στη νίκη σε περίπτωση ήττας. Αυτή η διαδικασία επαναλαμβάνεται 100 φορές και θεωρούμε ως long term evaluation το μέσο όρο των τιμών που λάβαμε μέσω της evaluation function μας.

Για να αποφασίσουμε αν θα ακολουθήσει μία από τις προτάσεις των συμπαικτών μας, αρχικά θεωρούμε την δική μας sequence of actions ως την καλύτερη δυνατή, την οποία αξιολογούμε με τους δύο τρόπους που αναφέρθηκαν παραπάνω. Έπειτα τη συγκρίνουμε με κάθε άλλο sequence που μας προτάθηκε, και αν οι long term και η short term αξιολογήσεις τους είναι μεγαλύτερες από τις αντίστοιχες που βρήκαμε για τη δικιά μας, θεωρούμε αυτήν ως τη βέλτιστη στρατηγική και επαναλαμβάνουμε τη διαδικασία για όλες τις προτάσεις.

Για την συνολική αξιολόγηση των συμπαικτών, αποθηκεύουμε σε ένα πίνακα μεταβλητές που συμβολίζουν το πόσο καλύτερες ή χειρότερες υπήρξαν οι προτάσεις των συμπαικτών μας από τη στρατηγική που αποφασίσαμε για ένα γύρο μέσω του MCTS. Χωρίζουμε το παιχνίδι σε δύο μέρη:

Early Stage:

Early stage θεωρούμε τους πρώτους (3*αριθμό παικτών) γύρους, δηλαδή μέχρι όλοι οι παίκτες να έχουν παίξει τρεις φορές. Η πρόβλεψη σε αυτό το στάδιο για το πώς θα καταλήξει το παιχνίδι μέχρι το τέλος είναι δυσκολότερη, και για αυτόν τον λόγο δίνουμε μεγαλύτερη έμφαση στη short term evaluation, με κάθε agent να λαμβάνει +10 reward για τις προτάσεις με καλύτερη short term evaluation από τη δικιά μας(-10 για χειρότερη). Με λιγότερη έμφαση στη long term, κάθε agent να λαμβάνει +5 reward για τις προτάσεις με καλύτερη long term evaluation από τη δικιά μας.

Late Stage:

Θεωρούμε το στάδιο που η πρόβλεψη για το πώς θα καταλήξει το παιχνίδι μέχρι το τέλος είναι πιο κοντινή στη πραγματικότητα. Για αυτόν τον λόγο δίνουμε μεγαλύτερη έμφαση στη long term evaluation, με κάθε agent να λαμβάνει +15 reward για τις προτάσεις με καλύτερο short term evaluation από τη δικιά μας(-15 για χειρότερη). Με λιγότερη στη short term, κάθε agent να λαμβάνει +5 reward για τις προτάσεις με καλύτερη short term evaluation από τη δικιά μας(-5 για χειρότερη).

Όταν το ολικό evaluation ενός πράκτορας ξεπεράσει το threshold των 55 πόντων, ακολουθούμε κάθε πρόταση του. Το threshold επιλέχθηκε εν μέρει αυθαίρετα γιατί δεν είχαμε την ευκαιρία να συγκρίνουμε τα λαμβανόμενα suggestions από καλύτερους από τον δικό μας πράκτορες(χρησιμοποιήθηκαν μονάχα παλαιότερες εκδόσεις του πράκτορα). Ταυτόχρονα προσπαθούμε να δίνουμε την ευκαιρία σε agents με καλύτερο long term evaluation να μας “κυριεύσουν”, καθώς αυτό αντιστοιχεί καλύτερα στο πραγματικό evaluation του παιχνιδιού.

Επίσης, σκεφτήκαμε την υλοποίηση ενός κατώτερου threshold για τους agents που οι προηγούμενες προτάσεις τους ήταν τόσο κακές, που δε χρειάζεται να τις λάβουμε υπόψη μας για την εύρεση του καλύτερου sequence. Όμως “a broken clock is right twice a day” και το υπολογιστικό κόστος, για την αξιολόγηση κάθε πρότασης, είναι αρκετά μικρό ώστε να μπορούμε να παραλείψουμε την θέσπιση ενός τέτοιου threshold.

Μελλοντική δουλειά/Βελτιώσεις

Μια βελτιωμένη έκδοση θα περιελάμβανε μια λεπτομερέστερη μοντελοποίηση των συμπαικτών μας με βάση τις παλαιότερες αποφάσεις τους στην υπάρχουσα αλλά και παλαιότερες παρτίδες. Επιπλέον θα μπορούσαμε να προσαρμόσουμε δυναμικά την στρατηγική μας σε σχέση με το προφίλ των συμπαικτών μας.

Βιβλιογραφία/Πηγές

[1]https://boardgamegeek.com/thread/407054/fun-point-point-maps-analysis-optimal-research-sta

[2] https://thefunnybrain.com/2018/12/28/board-game-strategy-tips-pandemic/

[3] https://www.youtube.com/watch?v=2lsF28EOrrg&feature=youtu.be

[4] https://www.baeldung.com/java-monte-carlo-tree-search

[5]https://www.analyticsvidhya.com/blog/2019/01/monte-carlo-tree-search-introduction-algorithm-deepmind-alphago/?fbclid=IwAR0ERpdQ-KgRj3rM1tQLw8G5oDBYwVFGr9WnT6Fgnm8e_EYpNafA1OVS064

[6] https://philippmuens.com/minimax-and-mcts

[6] https://webdocs.cs.ualberta.ca/~mmueller/ps/2013/2013-gochapter-preprint.pdf